REALbasic 2005 - What's New

DNS And E-mail

MacTech Magazine
July • 2005

MACTECH®

The Journal of Macintosh Technology

Tiger Server At Your Service

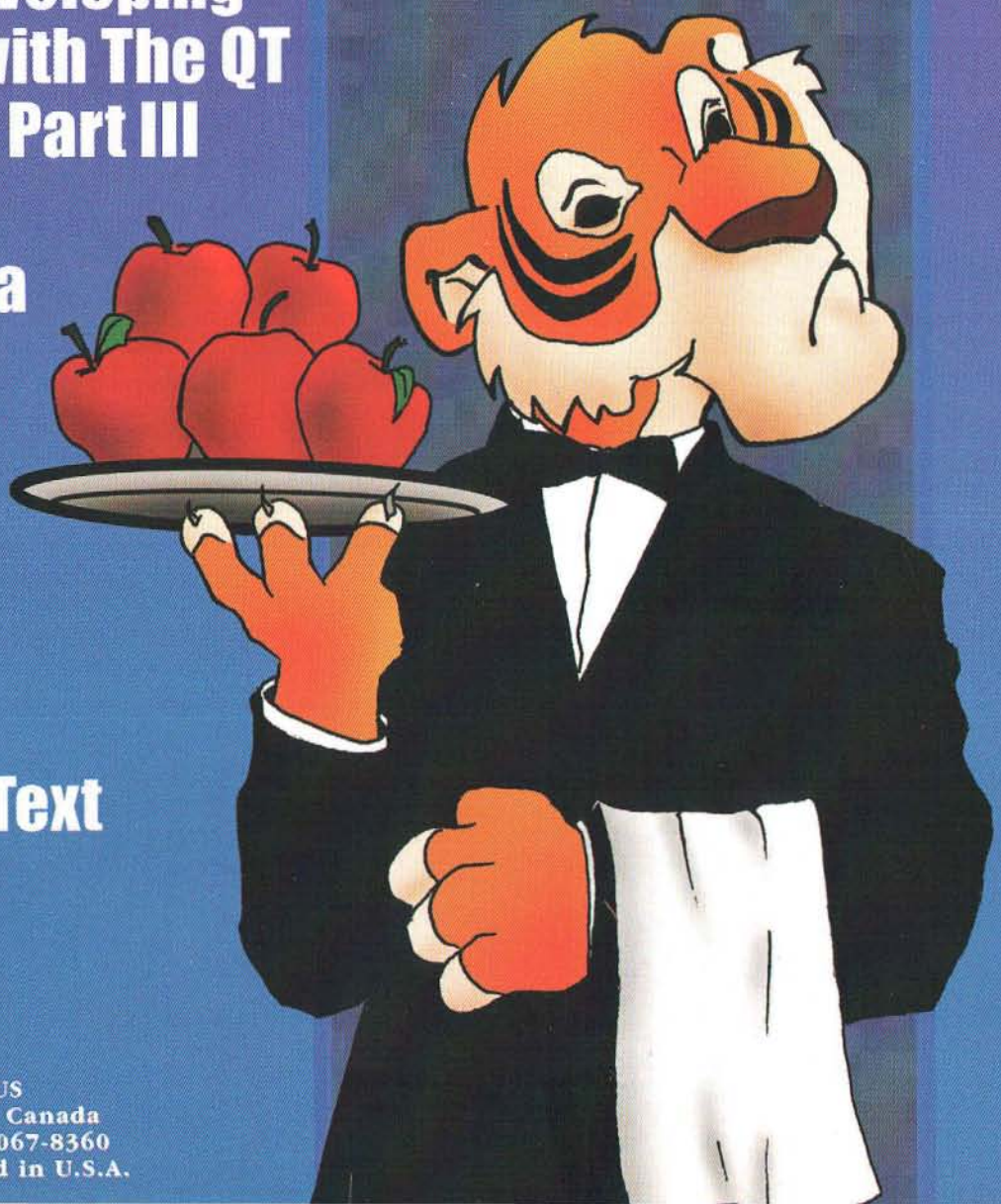
By John Welch

QuickTime: Developing Applications with The QT For Cocoa Kit - Part III

Style For Cocoa Programmers

Introduction To Core Data

AppleScript - Working with Text



07
0 32128 74887 8

$8.95 US
$12.95 Canada
ISSN 1067-8360
Printed in U.S.A.

Need to find something fast?

INDEX INDEX INDEX

With c-tree Speed.

TRY IT NOW!

FairCom's c-tree Plus®

embedded database engine offers Superior Indexing Technology – the key to performance, data integrity, and concurrency. c-tree Plus offers direct record-oriented C and C++ APIs with an industry-standard SQL interface that allows use of any combination of APIs within the same application. Furthermore, we offer source code access for intimate programming control, unmatched portability, and developer-to-developer technical support.

Heterogeneous Environments

c-tree Plus and c-treeSQL™ Servers are the perfect solution for your mixed platform environments. Mac servers to Windows clients? No problem. Linux Servers to Mac clients? No problem. c-tree has a long history of cross-platform development solutions. Byte incompatibility between platforms is handled seamlessly with our Uniformat data handling technology.

Low TCO

c-tree is priced affordably, requires minimal hardware resources, and needs no IT staff for maintenance. If Total Cost of Ownership (TCO) is important to you, c-tree is the perfect database.

Easy Deployment

c-tree Servers are designed for ease of use and deployment as well. Out of the box, our servers can be installed and running in minutes.

Start indexing your data today!

FairCom®

USA • Europe • Japan • Brazil

www.faircom.com

Go to www.faircom.com/go/mteval for a FREE evaluation of c-tree Plus!

Getting Started

By Dave Mark

iClip And Advanced UNIX Programming

So guess what? I got a boat. OK, it's relatively small as far as boats go, but it gets me from the shores of beautiful Lake Anna to the islands, beaver dams, and fishing holes that dot this lovely retreat. PowerBook in hand, floating along, trailing my feet in the water. Ah, this is the life!

Um, Dave, you are supposed to be writing a column. Oh, yes. As you can tell, I am on vacation this month. Rather than skipping this column entirely, I wanted to write about two really cool products I've been playing with lately. The first is a book, the second an innovative clipboard program.

Advanced Programming

Laurie Anderson once said, "You know, I could write a book. And this book would be thick enough to stun an ox." I believe she was talking about the brand new, second edition of W. Richard Stevens and Stephen A. Rago's *Advanced Programming in the UNIX Environment*. This is an amazing book. Yes, it is indeed thick enough to stun an ox. But even better, it is chock full of POSIX-compliant sample code that walks you through all manner of Unix programming techniques.

You can download a tar file with all the source code and makefiles from ftp.uu.net. Login using anonymous FTP and grab the file *published/books/stevens.advprog.tar.Z*. I really love the sample code in the book. I just wish the author thought enough of the Mac to provide an Xcode-savvy version of the sample code. As is, you'll need to do a bit of tweaking to get the examples to compile using Xcode. Or you can just use the book as reading material to get a sense of how Unix really works. There is a *lot* here. That said, the authors do discuss Mac OS X and Darwin and have tested the book's code on a Mac.

The book starts off with a Unix System Overview, covering the basic architecture, logins, shells, files and directories, process control, threading, etc. The first sample

MACTECH

A publication of
XPLAIN CORPORATION

TABLE OF CONTENTS

DEPARTMENTS

FEATURED ARTICLES

KOOL TOOLS

program is a bare-bones implementation of the *ls* command. This is followed by a pair of file-copying programs, a simple program that prints its program ID, and a simple shell. This is all in chapter 1.

Chapter 2 features a terrific discussion of UNIX standardization and implementations. I found this chapter fascinating, with its discussion of the evolution of ISO C, IEEE Posix, through the Single UNIX Specification.

There is just *so* much richness in this book. So much to learn. Wonderful information on file i/o, files and directories, process control, race conditions, v-node tables, job control, signals, etc., etc., etc.

If you are interested in what makes UNIX tick, this book is a must have.

Addison-Wesley, $74.99,
http://www.awprofessional.com/title/0201433079

Inventive iClip

There are a number of products out there that I like to call "desktop information managers". They are not PIMs (personal info managers), not classic databases. Instead, they live either on the desktop or in the shadowy universe behind and between your mainstream applications. Classic example of this are Launch Bar 4 (http://www.obdev.at/products/launchbar/), Butler (http://www.petermaurer.de/nasi.php?thema=butler&sprache=english), TigerLaunch (http://ranchero.com/tigerlaunch/), and oh so many other utilities that promise to organize your files and make it easier to launch your applications.

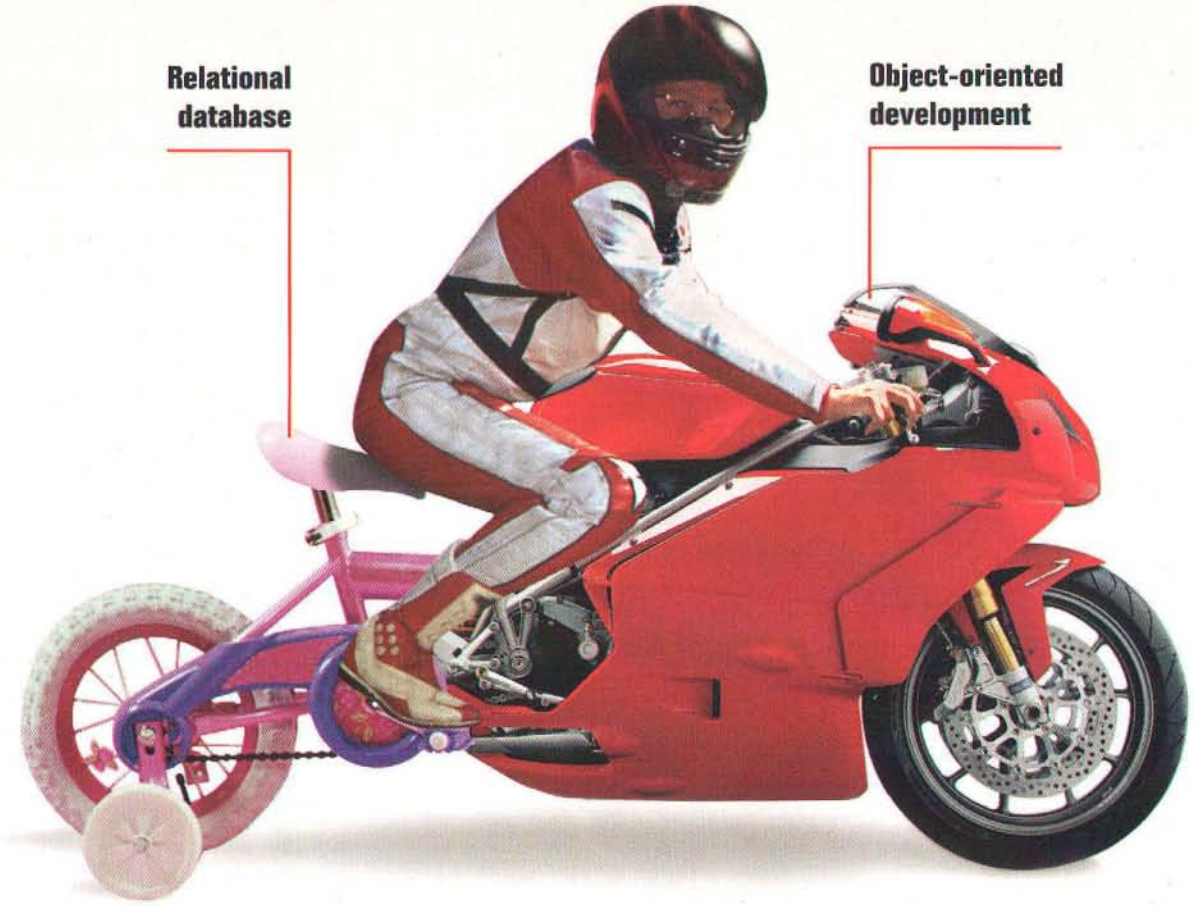
Another category is the list managers, utilities that help you organize your data, allowing you to build collections of bookmarks, URLs, etc. It's becoming a bit difficult to distinguish between all these products, as they are all becoming somewhat similar, borrowing features from each other. Launchers let you launch URLs. Organizers let you organize files and launch them.

It is a bit refreshing when an entirely original idea floats into view. iClip, from Inventive, is just such a creature. On the surface, iClip looks like just another clipboard management utility. You can drag data from your app to iClip's main floating window, and the data is stored in a cell within the window. Want the data again, just drag it back out.

The window can be formatted vertically or horizontally. Hover over one of the cells and a cartoon balloon appears, showing a bit more detail about that data (see Figure 1).

MacTech Magazine (ISSN: 1067-8360 / USPS: 010-227) is published monthly by Xplain Corporation, 850-P Hampshire Road, Westlake Village, CA 91361-2800. Voice: 805/494-9797, FAX: 805/494-9798. Domestic subscription rates are $47.00 per year. Canadian subscriptions are $59.00 per year. All other international subscriptions are $97.00 per year. Domestic source code disk subscriptions are $77 per year. All international disk subscriptions are $97.00 a year. Please remit in U.S. funds only. Periodical postage is paid at Thousand Oaks, CA and at additional mailing office.

POSTMASTER: Send address changes to **MacTech Magazine**, P.O. Box 5200, Westlake Village, CA 91359-5200.



A BETTER DATABASE CAN SPEED UP YOUR DEVELOPMENT CYCLE

If your relational database isn't a good match for your object-oriented development, you need a new database.

Caché, the *post-relational* database from InterSystems, combines high-performance SQL for faster queries and an advanced object database for rapidly storing and accessing objects. With Caché, no mapping is required between object and relational views of data. That means huge savings in both development and processing time.

Applications built on Caché are massively scalable and lightning fast. They require little or no database administration.

More than just a database system, Caché incorporates a powerful Web application development environment that dramatically reduces the time to build and modify applications.

Caché is so reliable, it's the world's leading database in healthcare – and it powers enterprise applications in financial services, government and many other sectors. With its high reliability, high performance and low maintenance, Caché delivers your vision of a better database.

We are InterSystems, a specialist in data management technology for over twenty-six years. We provide 24x7 support to four million users in 88 countries. Caché is available for Windows, OpenVMS, MAC OS X, Linux, and major UNIX platforms – and it is deployed on systems ranging from two to over 50,000 simultaneous users.

InterSystems
CACHÉ
Make Applications Faster

Try a better database. For free.

Download a free, fully functional, non-expiring copy of Caché or request it on CD at www.InterSystems.com/match3

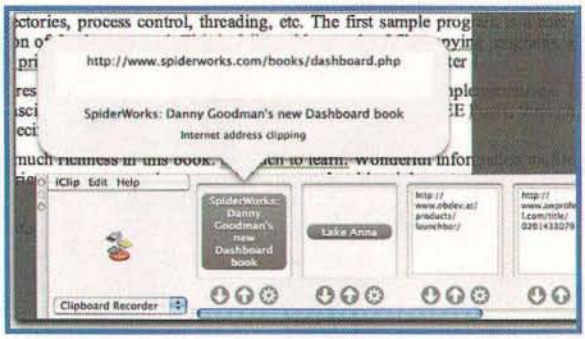


Figure 1. The iClip window, with an info cartoon balloon showing a bit more detail.

You can organize your clippings into sets and easily switch between sets. Yes, yes, yes, this all sounds familiar. We've seen this kind of organizer in many different forms over the years, from multi-clip to any one of the current crop of clipboard utilities.

But.

iClip offers one feature that really caught my eye. You can set iClip up as a clipboard recorder. In this mode, iClip catches every single copy of data to the clipboard and adds that data to its own cell in the clipboard recorder window. Each time you copy, that data hangs around, even if you move between different applications. You can hide the iClip window and the clipboard gathering continues. Want to retrieve something you copied a while back? Hit the iClip hot-key and fish your data from the clipboard recorder. Ah, good, it's still there!

This might sound small, but I love this feature. There *are* other products that track clipboard progress. I just like the way iClip does it. Give it a try. You can download iClip from the Inventive web site:

http://inventive.us/iClip/

Until Next Month

Next month, I'll be back full force with my regular column. Until then, enjoy your summer. I'll be here on the lake, counting the clouds. ☺

About The Author

Dave Mark is a long-time Mac developer and author and has written a number of books on Macintosh development. Dave has been writing for MacTech since its birth! Be sure to check out the new Learn C on the Macintosh, Mac OS X Edition at http://www.spiderworks.com.

MACTECH®

Communicate With Us

DEPARTMENT E-mails

Orders, Circulation, & Customer Service
cust_service@mactech.com

Press Releases
press_releases@mactech.com

Ad Sales
adsales@mactech.com

Editorial
editorial@mactech.com

Online Support
online@mactech.com

Accounting
accounting@mactech.com

Marketing
marketing@mactech.com

General
info@mactech.com

Web Site
http://www.mactech.com

In this electronic age, the art of communication has become both easier and more complicated. Is it any surprise that we prefer **e-mail?**

If you have any questions, feel free to call us at 805/494-9797 or fax us at 805/494-9798.

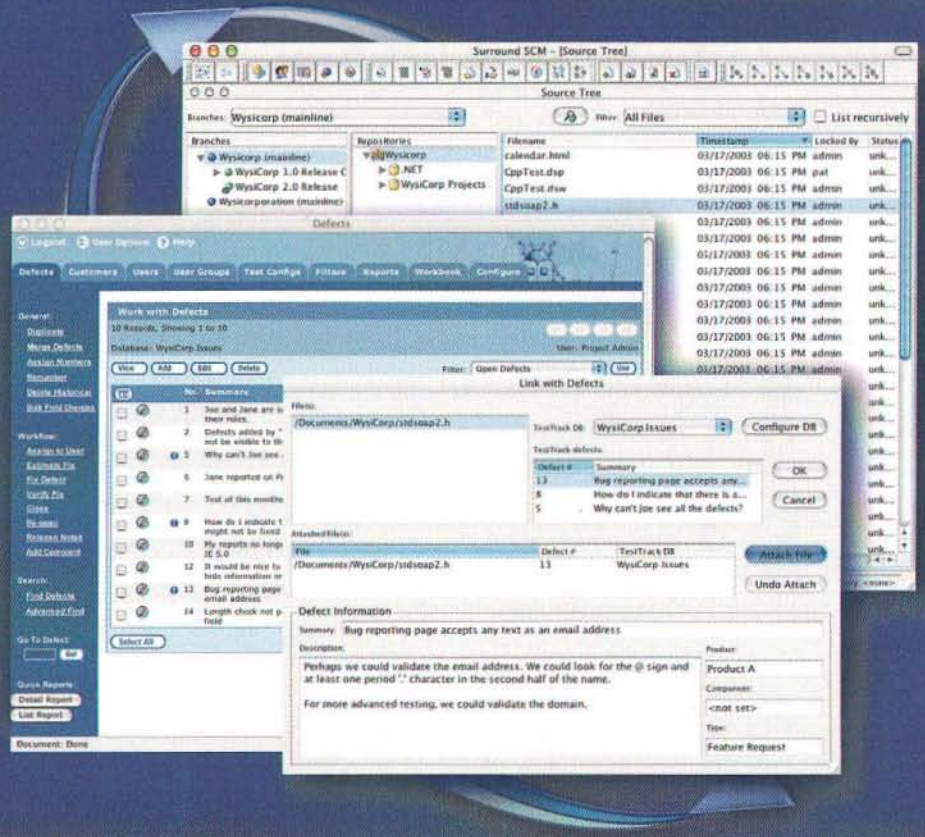
If you would like a subscription or need customer service, feel free to contact MacTech Magazine Customer Service at 877-MACTECH

We love to hear from you! Please feel free to contact us with any suggestions or questions at any time.

Write to letters@mactech.com or editorial@mactech.com as appropriate.

Complete Source Control and Defect Management for Mac OS X

Seapine Software™
changing the world of software development



Effective source code control and defect tracking require powerful, flexible, and easy-to-use tools—Surround SCM and TestTrack Pro

- Complete source code control with private workspaces, automatic merging, role-based security, and more
- Comprehensive defect management — track bug reports and change requests, define workflow, customize fields
- Fast and secure remote access to your source files and defects — work from anywhere
- Advanced branching simplifies managing multiple versions of your products
- Link code changes with defects and change requests — know who changed what, when, and why
- Scalable and reliable cross-platform, client/server solutions support Mac OS X, Windows, Linux, and Solaris
- Exchange data using XML and ODBC, extend and automate with SOAP support
- Licenses priced to fit your budget

Seapine Software Product Lifecycle Management
Award winning, easy-to-use software development tools

Seapine Surround SCM
Seapine TestTrack PRO

software development 13th annual product excellence award

Download Surround SCM and TestTrack Pro at
www.seapine.com
or call 1-888-683-6456

BACK TO THE FUTURE, PART III

DEVELOPING APPLICATIONS WITH THE QUICKTIME FOR COCOA KIT

In the previous two *QuickTime Toolkit* articles (in *MacTech*, May and June 2005), we've taken a look at using QTKit, Apple's new Cocoa framework for displaying and modifying QuickTime movies. We saw how to open and manipulate movies using command-line tools; then we stepped through the process of building a fairly complete document-based Cocoa application using QTKit. This application, which we called KitEez, supports the standard document-related behaviors, including saving an edited movie into its original file or into some other file, and reverting to the last saved version of a movie file. Since KitEez was built using the Cocoa class NSDocument, we didn't need to write any code to track the edited state of the document window and its associated movie or to display the standard dialog boxes that warn the user about unsaved changes and overwriting existing files.

Introduction

In this article, we'll take a look at a few more of the capabilities offered in by QTKit. In particular, we'll see how to add images to an existing QuickTime movie and work with notifications and delegates. We'll also see how to call QuickTime functions that have no corresponding QTKit method and we'll take a second look at opening files and URLs.

Adding Images to a Movie

Adding Images to an Existing Movie

Imagine that we've opened a QuickTime movie from a file and we'd like to tack a few images onto the end of the movie. The QTMovie class provides a method that makes this extremely easy, addImage:forDuration:withAttributes, whose signature is this:

```
-(void)addImage:(NSImage *)image
        forDuration:(QTTime )duration
        withAttributes:(NSDictionary *)attributes;
```

The image parameter of course specifies the image to add to the end of the movie, and the duration parameter is a QTTime structure that specifies the desired duration of the new frame. The attributes parameter is a dictionary that specifies the codec to be used to compress the image and possibly also the quality of the compressed image. For instance, Listing 1 shows a simple method that reads an image from the application's nib file and adds it to the end of the movie, with a duration of one second.

Listing 1: Appending an image to a movie

```
- (void)addImageFromNibFile:(NSString *)name
{
  NSImage *image = [NSImage imageNamed:name];
  QTTime duration = QTMakeTime(600, 600);
```

```
    NSDictionary *dict = [NSDictionary dictionaryWithObject:
        QTStringForOSType('pxlt') forKey:QTAddImageCodecType]

    [_movie addImage:image forDuration:duration
                      withAttributes:dict];
}
```

As you can see, the dict dictionary contains one key-value pair, where the key is QTAddImageCodecType. This key specifies the codec to be used to compress the image before it's added to the movie. Notice that we have used the function QTStringForOSType to convert a codec OSType (here, for the Pixlet compressor) to an NSString object. QTKit also provides a utility function for going in the opposite direction: QTOSTypeForString will return an OSType corresponding to a given NSString.

The dictionary of attributes can also contain a key-value pair in which the key is QTAddImageCodecQuality. If this pair is present, then the specified value, which should be an NSNumber, is used as the quality setting for the compressor. This key-value pair is optional; if it is not included in the dictionary, the quality setting codecNormalQuality is used. The acceptable values are listed in this enumeration in the header file ImageCompression.h in the framework QuickTime.framework:

```
enum {
  codecLosslessQuality          = 0x00000400,
  codecMaxQuality               = 0x000003FF,
  codecMinQuality               = 0x00000000,
  codecLowQuality               = 0x00000100,
  codecNormalQuality            = 0x00000200,
  codecHighQuality              = 0x00000300
};
```

Adding Images to a New Movie

Suppose that instead of adding an image to the end of an existing movie, we want to create a slideshow movie from some images, starting this time from an empty movie. You might think that we could first create a new, empty QTMovie object and then tack on the images, like this:

```
QTMovie *movie = [[QTMovie alloc] init];
[movie addImageFromNibFile:@"A.jpg"];
[movie addImageFromNibFile:@"B.jpg"];
// and so forth...
```

In fact, however, this would fail. The underlying reason is that a new QTMovie object created in this way has no writable data reference associated with it. In layman's terms, this means that the associated QuickTime Movie has not yet been assigned a place to store any new media data that is added to the movie. Since there is no place to write the media data, the call to addImage:forDuration:withAttributes fails.

You might think that this is a bug in QTMovie's init method, which could easily assign a block of data in memory as the writable data reference for a new QTMovie object. But that revised behavior would lead to problems of its own. For instance, adding a number of large images could soon fill up the available physical memory or at least put the virtual memory system into severe thrashing.

Future versions of QTKit will almost certainly provide one or more new methods in QTMovie to create a new empty QTMovie object that is associated with a writable file or block of memory. In the meantime, it is possible to work around this limitation by creating a new empty file and associating it with a new QTMovie object using the movieWithQuickTimeMovie:disposeWhenDone:error: method, as shown in Listing 2. As you can see, we use the *movie storage* APIs (discussed in "Modern Times" in *MacTech*, May, 2004) to create a new empty movie file.

Listing 2: Creating a new empty movie that is writable

```
- (QTMovie)movieWithWritableFile:(NSString *)filename
                         handler:(DataHandler
*)dataHandler
{
  OSErr err = noErr;
  Handle dataRef = nil;
  OSType dataRefType;
  Movie qtMovie = NULL;
  QTMovie movie = nil;

  // make sure we are passed a location to return the data handler
identifier
  if (!dataHandler) return nil;

  // create a file data reference for the specified filename
  err = QTNewDataReferenceFromFullPathCFString(
              (CFStringRef)fileName,
              kQTNativeDefaultPathStyle,
              0, &dataRef, &dataRefType);
  if (err != noErr) return nil;

  // create a QuickTime movie from the file data reference
  err = CreateMovieStorage(dataRef, dataRefType,
'TVOD',
              smSystemScript, newMovieActive,
              dataHandler, &qtMovie);
  if (err != noErr) return nil;

  // instantiate a QTMovie from the QuickTime movie
  movie = [QTMovie movieWithQuickTimeMovie:qtMovie
              disposeWhenDone:YES error:nil];

  // mark the movie as editable
  [movie setAttribute:[NSNumber numberWithBool:YES]
              forKey:QTMovieEditableAttribute];

  return movie;
}
```

We need to return to the caller both the new QTMovie object and the identifier of the data handler associated with the open movie file. That's because, when we are finished adding frames to the movie and have called updateMovieFile, we need to call CloseMovieStorage to close the movie:

```
CloseMovieStorage(dataHandler);
```

Creating a Movie from a Single Image

It's worth mentioning one further twist on this issue, namely using QTKit to create a movie from a single image. Suppose we want to create a 10-second movie

from a single image. We could use the strategy employed in the previous section, first creating a new empty movie file and then adding the specified image to its associated QTMovie object. But there is in fact a much simpler way to do this, using the dataReferenceWithReferenceToData:name:MIMEType: method in the QTDataReference class. In a nutshell, this method creates a data reference to some block of memory addressed using an NSData object and optionally attaches a filenaming extension or a data reference extension of type 'mime' to the data reference. As we have seen in earlier articles (particularly in "Somewhere I'll Find You" in *MacTech*, October, 2000), these extensions help QuickTime find the appropriate movie or graphics importer when a data reference is to a block of memory.

In the present case, we'll create an NSData object that contains a TIFF representation of the image and then create a QTDataReference object that has, as its filenaming extension, an arbitrary name with the extension .tiff. Listing 3 shows the code we can use to do this.

Listing 3: Creating a movie from a single image

```
- (void)createMovieFileFromImage:(NSString *)filename
                        (NSImage *)image
{
  if (!filename || !image)
    return;

  NSData *data = [image TIFFRepresentation];
  QTDataReference *dataRef = [QTDataReference
          dataReferenceWithReferenceToData:data
          name:@"some_image.tiff" MIMEType:nil];
  QTMovie *movie = [QTMovie movieWithDataReference:dataRef
          error:nil];

  // make the movie editable
  [movie setAttribute:[NSNumber numberWithBool:YES]
              forKey:QTMovieEditableAttribute];

  // set the duration to 10 seconds
  QTTimeRange range = QTMakeTimeRange(QTZeroTime,
                                      [movie duration]);
  [movie scaleSegment:range newDuration:QTMakeTime(10, 1)];

  // export as a 3gpp file
  NSDictionary *dict= [NSDictionary
        dictionaryWithObject:[NSNumber numberWithBool:YES]
        forKey:QTMovieFlatten];
  [movie writeToFile: filename withAttributes:dict];
}
```

This works because we are not actually editing the movie's media data. Instead, we are creating a QTMovie object that gets its media data from the NSData block. We *are* editing the movie when we call the scaleSegment:newDuration: method, but that's okay even when the movie does not have a writable data reference.

QuickTime Functions

Even though QTKit exposes a fairly large number of classes and methods, and even though it does quite a bit of movie-related processing automatically, it's fairly likely that we will want to use QuickTime capabilities that it does not currently support. For instance, we've already seen that QTKit does not yet provide a method for creating a new, empty movie that has a writable data reference associated with it. So we used the underlying QuickTime APIs to create one, which we then used to initialize a QTMovie object. It can also happen that we've already got a QTMovie object and we want to perform some operation not yet supported by QTKit. In this case, the QTMovie class provides two useful methods that allow us to retrieve the Movie and MoveController identifiers associated with a QTMovie object:

```
-(Movie)quickTimeMovie;
-(MovieController)quickTimeMovieController;
```

Suppose, for example, that we want to set the magnification (or "zoom") level of a Flash movie. We can do so by using the quickTimeMovie method to retrieve the QuickTime Movie associated with a QTMovie object and then using Movie Toolbox and Flash media handler functions, as shown in Listing 4.

Listing 4: Setting the zoom level of a Flash movie

```
- (void)setZoom:(float)zoomPct
{
  Track flashTrack = NULL;
  Media flashMedia = NULL;
  MediaHandler flashHandler = NULL;

  flashTrack = GetMovieIndTrackType([self quickTimeMovie],
                1, FlashMediaType, movieTrackMediaType |
                movieTrackEnabledOnly);
  if (flashTrack) {
    flashMedia = GetTrackMedia(flashTrack);
  flashHandler = GetMediaHandler(flashMedia);
    FlashMediaSetZoom(flashHandler, zoomPct);
  }
}
```

In general, it should be safe to call virtually any QuickTime functions on the Movie or the MoveController associated with a QTMovie object. Some operations, however, might not be safe and should probably be avoided. In particular, QTKit generally assumes that it will be disposing of the Movie and MovieController associated with a QTMovie object, so you should not call DisposeMovie or DisposeMovieController. (The exception to this rule is when you pass the value YES as the disposeWhenDone parameter in QTMovie's movieWithQuickTimeMovie:disposeWhenDone:error: method, which tells QTKit that you want to dispose of the movie yourself.) Also, deleting tracks from a movie using the Movie Toolbox function DisposeMovieTrack is likely to confuse QTKit and may even lead to crashes. It's likely that QTKit will in future versions add methods that provide a way to delete tracks, since this is a reasonably common operation for some kinds of applications.

Notifications and Delegate Methods

As you no doubt already know, a *notification* is a way for one Cocoa object to inform other objects about changes in its state or its properties. The QTMovie class defines a large number

of notifications that other objects can listen for; here are the notifications that are currently defined:

```
NSString *QTMovieEditabilityDidChangeNotification;
NSString *QTMovieEditedNotification;
NSString *QTMovieLoadStateDidChangeNotification;
NSString *QTMovieLoopModeDidChangeNotification;
NSString *QTMovieMessageStringPostedNotification;
NSString *QTMovieRateDidChangeNotification;
NSString *QTMovieSelectionDidChangeNotification;
NSString *QTMovieSizeDidChangeNotification;
NSString *QTMovieStatusStringPostedNotification;
NSString *QTMovieTimeDidChangeNotification;
NSString *QTMovieVolumeDidChangeNotification;
NSString *QTMovieDidEndNotification;
NSString *QTMovieChapterDidChangeNotification;
NSString *QTMovieChapterListDidChangeNotification;
NSString *QTMovieEnterFullScreenRequestNotification;
NSString *QTMovieExitFullScreenRequestNotification;
NSString *QTMovieCloseWindowRequestNotification;
```

Most of these are fairly obvious. For instance, QTMovieDidEndNotification is posted when a QTMovie object reaches its end. And QTMovieEditedNotification is posted when a QTMovie object has been edited or changed in some way. A few of these are however somewhat less obvious. The QTMovieTimeDidChangeNotification notification is not, for instance, posted whenever the movie time changes; rather, it's posted whenever the movie time changes to a value different from what it would be during normal movie playback. These sorts of time changes include operations like the user clicking in the movie controller bar to change the movie time or a wired action setting the movie time to some specific time.

In most cases, these notifications are posted using the NSNotificationCenter method postNotificationName:object:, where no accompanying userInfo dictionary is passed. The expectation is that the notification listener will be able to retrieve whatever information about the QTMovie object is needed at the time the notification is received. So, for instance, a listener receiving the QTMovieVolumeDidChangeNotification notification could call the QTMovie method volume to retrieve the current volume of the movie. In three cases, however, a dictionary of information is passed along with the notification, because there is no easy way for the receiver to ask for that information; these are:

```
QTMovieMessageStringPostedNotification
QTMovieRateDidChangeNotification
QTMovieStatusStringPostedNotification
```

We've already seen an example of registering for notifications in the previous article. When we initialize a movie view to hold a movie, we want the associated document to be informed of any size changes that occur programmatically or via wired actions; we did that by registering for the QTMovieSizeDidChangeNotification notification, like this:

```
[[NSNotificationCenter defaultCenter] addObserver:self
      selector:@selector(boundsDidChange:)
      name:QTMovieSizeDidChangeNotification object:movie];
```

Listing 5 shows our implementation of the boundsDidChange: method, which is invoked when we receive this notification.

Listing 5: Handling size-changed notifications

```
- (void)boundsDidChange:(NSNotification *)notification
{
  // set the size of the movie window to exactly enclose the movie at its
current size
  NSSize size = [[[movieView movie] attributeForKey:
                  QTMovieCurrentSizeAttribute] sizeValue];

  [[movieView window] setContentSize:
                  [self windowContentSizeForMovieSize:size]];
}
```

Notifications provide a very loose sort of coupling between objects in Cocoa. One object posts a notification and one or more other objects can listen for that notification and respond appropriately. By contrast, a much tighter association of two objects can be formed by designating one of them as the *delegate* of the other. Currently QTKit defines only five public delegate methods, all in the QTMovie class, of which only three are likely to be of use to developers:

```
- (BOOL)movie:(QTMovie *)movie linkToURL:(NSURL *)url;
- (QTMovie *)externalMovie:(NSDictionary *)dictionary;
- (BOOL)movie:(QTMovie *)movie
              shouldContinueOperation:(NSString *)op
              withPhase:(QTMovieOperationPhase)phase
              atPercent:(NSNumber *)percent
              withAttributes:(NSDictionary *)attributes;
```

A delegate's movie:linkToURL: method is called when the movie controller is about to open a movie specified by a URL. (This corresponds to the movie controller processing an mcActionLinkToURL action in the Carbon world.) For most applications, the QTKit's normal processing of URLs is fine for most applications; you would define this delegate method if you wanted to reroute URLs to some other location or cancel URL opening altogether (by returning NO).

A delegate's externalMovie: method is called when the movie controller needs to find a movie external to some given movie, most often to send that other movie a wired action. Once again, QTKit contains code to find external movies and most applications can rely on that automatic processing.

The only QTMovie delegate method likely to be defined by applications is movie:shouldContinueOperation:withPhase:atPercent:withAttributes:, which provides a Cocoa wrapper for a movie progress procedure. Currently this is useful only when calling the writeToFile:withAttributes method, though it's possible that more operations in QTMovie will support this delegate method in the future.

Opening Movies

In our first QTKit article (mentioned earlier), we opened a movie specified by a filename using the initWithFile:error: method, like this:

```
QTMovie *movie = [QTMovie initWithFile:filename error:nil];
```

We can also open a movie specified by a URL using the initWithURL:error: method, like this:

```
QTMovie *movie = [QTMovie initWithURL:url error:nil];
```

It's useful to know that there is a more general method for opening movies which offers several advantages over these methods, namely initWithAttributes:error:. The following lines of code do exactly the same thing as the line of code that uses initWithFile:error: above.

```
NSDictionary *dict = [NSDictionary dictionaryWithObject:
      filename forKey:QTMovieFileNameAttribute]
QTMovie *movie = [QTMovie initWithAttributes:dict
                                error:nil];
```

The idea is that we pass into initWithAttributes:error: a dictionary containing one or more key-value pairs that specify the attributes we want the new QTMovie object to have. One of those attributes must specify the location of the movie data, either via a filename or a URL or a QTDataReference object or a pasteboard or an NSData block. But there can be an indefinite number of other attributes in that dictionary, which are applied to the movie object before it's returned to the caller. For instance, we can ensure that all movies we open are editable by creating the dictionary like this:

```
dict = [NSDictionary dictionaryWithObjectsAndKeys:
  filename, QTMovieFileNameAttribute,
  [NSNumber numberWithBool:YES], QTMovieEditableAttribute,
  nil];
```

The initWithAttributes:error: method is not merely a convenience that saves us from setting attributes on the QTMovie object returned by calls to initWithFile:error: or initWithURL:error:. There are at least two important reasons we might need to call it rather than those other methods. First, we can pass in attributes that override some of the default behaviors of QTKit when opening movies. For example, we learned in an earlier article that QTKit opens all movies asynchronously. (That is, a call to initWithURL:error: returns almost immediately, so that the application can continue with its processing while the movie data loads.) But it's possible that an application would want to load movies synchronously. In that case, if could construct an attributes dictionary like this:

```
dict = [NSDictionary dictionaryWithObjectsAndKeys:
  url, QTMovieURLAttribute,
  [NSNumber numberWithBool:NO], QTMovieOpenAsyncOKAttribute,
  nil];
```

Setting NO as the value of the QTMovieOpenAsyncOKAttribute attribute indicates that we want the movie to be opened synchronously. Obviously, this attribute needs to be set *before* we start opening the remote movie, and initWithAttributes:error: is the only way to do that using QTKit methods.

The second case in which this method is sometimes necessary is to set a delegate for the QTMovie object being created so that that delegate is operational during the movie loading. In particular, it's quite possible that an mcActionLinkToURL action is encountered by the movie controller before the initWithURL:error: method returns to the caller. So this sequence of calls might not work if the delegate needs to handle all link-to-URL actions:

```
QTMovie *movie = [QTMovie initWithURL:url error:nil];
[movie setDelegate:self];
```

Instead, we can call initWithAttributes:error:, adding the QTMovieDelegateAttribute and its associated object to the dictionary of attributes:

```
dict = [NSDictionary dictionaryWithObjectsAndKeys:
  url, QTMovieURLAttribute,
  self, QTMovieDelegateAttribute,
  nil];
QTMovie *movie = [QTMovie initWithAttributes:dict
                                error:nil];
```

By setting the delegate in this way, we can guarantee that all link-to-URL actions will be seen by the movie:linkToURL: delegate method.

If your application does not need to use any of the delegate methods defined in the QTMovie class and if it does not need to override any of the default behaviors provided by QTMovie, then the initWithFile:error: and initWithURL:error: methods are just fine.

Conclusion

This article and the previous two articles should help convince you that QTKit is a powerful framework that allows you to develop robust QuickTime applications with a minimum of code. QTKit provides the most comprehensive set of built-in capabilities and the most complete automatic processing of any QuickTime framework that I have yet encountered. It's easy to use, it dovetails nicely with existing Cocoa frameworks, and it's already serving as the QuickTime underpinnings of powerful in-house and commercial multimedia applications.

In the next article, we'll take one more look at QTKit, to investigate how to execute QTKit methods on a secondary thread. That will give us the machinery we need to perform computationally intensive operations (like exporting large movies or creating slideshow movies from a large number of images) without negatively impacting the responsiveness of the application's user interface.

References

The movieWithWritableFile:handler: method shown earlier is based on sample code that is available at http://developer.apple.com/samplecode/QTKitCreateMovie/QTKitCreateMovie.html.

About The Author

Tim Monroe is a member of the QuickTime engineering team at Apple. You can contact him at monroe@mactech.com. The views expressed here are not necessarily shared by his employer.

$32 < 1 Year

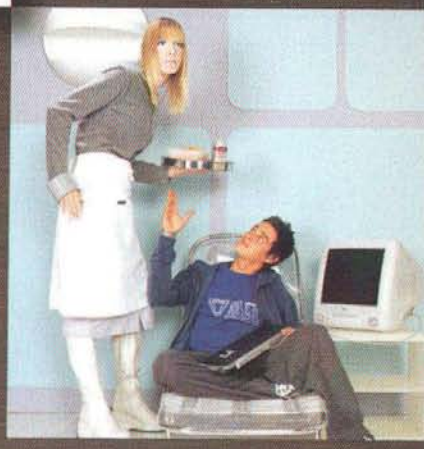
2 Years > $64

Interviews

Tapping into the world of celebrities and their Macs, only MacDirectory offers exclusive interviews. Get a close and personal view from Sarah Jessica Parker, Steve Jobs, Madonna, Harry Connick Jr., George Lucas, Jennifer Jason Leigh, Steve Woz and other leaders in the Mac community.

Features

Designers, writers, musicians, business leaders & our technical expert team offer their own personal interpretation of things that only the Mac system can deliver. With more than 200 pages of news, insights, trends and the largest Macintosh buyer's guide including over 5,000 Mac products and services.



MacDirectory

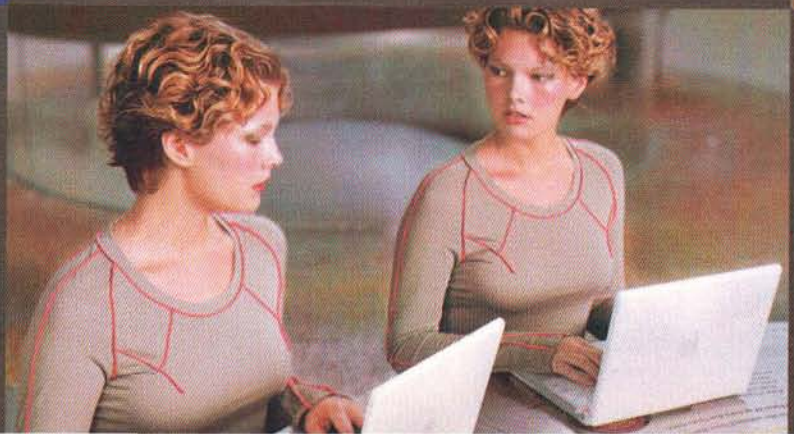
BEYOND ANY MACINTOSH MAGAZINE.

Culture

MacDirectory takes you to the wildest corners of the world and uncovers how Macintosh computers are being used by other cultures. Travel to Japan, Australia, Germany, Brazil & Russia and learn more about Apple's cultural impact around the globe.

Reviews

Find out all you need to know about the latest Mac products including the hottest Mac OS software and hardware.



Subscribe >

macdirectory.com

Send check or money order to:
MacDirectory Subscription Dept.
326 A Street, 2C
Boston, MA 02210

Introduction to Core Data

Using Tiger's New Persistence Framework and Modeling Tool

By Jeff LaMarche

Introduction

This our second article on the new Cocoa technologies available with the release of Tiger. In the last installment, we focused on the new tree-based XML parser, NSXML. This month and in future articles, we'll be delving into Core Data. In order to better understand what Core Data is and where it came from, a bit of history is in order.

Way back in the day, NeXT had a wonderful something called EOF: the Enterprise Objects Framework. EOF was not just a framework , but also a tool (called EOModeler) that allowed you to create your application's data model visually instead of manually creating objective-C classes. Even though your data model was not made by writing code, you could interact with your data as objective-C objects by using key-value coding; the framework handled all the work involved in persisting the data to a SQL database, flat file, or any other data store for which somebody wanted to write an EOAdaptor. You rarely had to write a single line of SQL or file management code in order to make your program work; all that happened for free.

EOF allowed unbelievable developer productivity in both creating and maintaining applications. And due to its object-oriented architecture, it offered limitless flexibility because you could always opt to write your own classes to override or supplement the functionality you got for "free" by using it. While a version of EOF still exists in WebObjects (EOF/WO), it is Java based and not available nor licensed for use in regular Cocoa applications. The objective-C version still exists and is installed with WebObjects, but it's not actively supported and the adaptors have not been kept up to date.

With the release of Tiger and Core Data, EOF has risen from the dead... sort of.

The More Things Change...

Apple's own documentation states that Core Data and EOF/WO share a common heritage. Despite that, they are clearly intended for different purposes. EOF/WO is used almost exclusively for writing database-backed web applications.

Core Data, on the hand, was specifically designed without support for remote databases; it is intended (at least for now) just to provide ***local*** data storage for Cocoa applications. Lack of support for remote databases, in the eyes of many developers, is a significant limitation. Despite that, Core Data does what it was designed to do admirably well. In some ways, Core Data is better than EOF. The modeling functionality is much more polished than what EOModeler offered and it's integrated right into XCode, so there's no need to switch back and forth between the IDE and modeller. Core Data's limited storage options also simplify things; the built-in storage options– which include storing data as XML, in binary files, or in an embedded SQLite database– means no more hassles finding the EOAdaptor you need or making sure your users have the correct adaptors (and version) installed on their machines.

Combined with Cocoa Bindings, which were added with Panther, Core Data is nothing short of miraculous in terms of the increase in developer productivity it affords you.

Overview and Terminology

The best way to show how wonderful Core Data is, is to just dive in and use it. Before we do that, however, let's take a second to go over some basic terminology.

Entity – this is the highest level object in a Core Data data model. Entities generally corresponds to the Cocoa classes traditionally used to hold data. They can hold data directly by way of their ***attributes*** and by incorporating other entities using ***relationships*** and ***fetched properties***. Programmatically, entities are most often represented by the NSManagedObject class, but can also be represented by custom subclasses of NSManagedObject. Entities can be abstract and Core Data supports a rudimentary form of inheritance among entities. This functionality is beyond the scope of this article, and (at least in the days of EOF) was seldom used in practice. In fact, at one time, the use of entity inheritance was discouraged in Apple's official documentation.

Attribute – an attribute is a property of an entity that holds data. Attributes roughly correspond to the instance variables of a traditional data model class excluding collections such as NSArray and instance variables that are themselves also custom data model classes. There are a limited number of attribute types, such as strings, a few types of numbers, dates and a few others. Attributes are represented in code by the class NSAttributeDescription.

Relationship – a relationship is Core Data's primary way of making one entity act as part of another entity. Relationships can be one-to-one—which is analogous to declaring an instance of one data model class as a variable of another class—or they can be one-to-many, which parallels the use of an NSArray instance variable. You use the NSRelationshipDescription to interact with relationships in your code.

Fetched Property – fetched properties are similar to relationships in that Fetched Properties allow objects to behave as if they are part of another object. The key difference is that while relationships tie specific entity instances to another entity, Fetched Properties incorporate entities based on specified criteria, very much like the rule-based "smart" functionality that has popped up in so many places lately: smart playlists in iTunes, smart mailboxes in Mail, smart photo albums in iPhoto, etc. NSFetchedPropertyDescription is the Cocoa class used to represent fetched properties.

The term "property" is used generically to refer to all items that can be contained with an entity: fetched properties, relationships, and attributes. Entities and the three types of properties make up the actual Core Data data model. Here are a few more Core Data terms and classes with which you should be familiar before we get our feet wet:

Context – the "context" is the virtual space in which data is stored and corresponds to the physical data store on your local hard drive. For a Core Data application, you'll typically only have one context, though you could have more. Within a Core Data Document-Based application, you'll typically have a separate context for each open document. In Core Data, the context is represented by the NSManagedObjectContext class.

Predicates – the easiest way to think of a predicate is to think of it as a "rule", in the sense of smart playlists. An example of a predicate, rendered in English, would be "name begins with 'A'" or "price is less than $20". The NSPredicate class and its two subclasses, NSCompoundPredicate and NSComparisonPredicate represent predicates in code. Predicates can be compounded using the logical operators and, or, and not.

Fetch Request – this is how you retrieve entities from the context. Fetch requests can be unqualified, in which case they will return all instances of a specific entity, or they can be qualified using predicates. Fetch Requests are represented by the NSFetchRequest class. You can also retrieve data from the context by using NSArrayControllers, which we'll do later in this article.

Creating the Data Model

You may remember in the NSXML article, we created a class called MTBook to store the data we retrieved from Amazon's XML-based web service. Today, we're going to recreate that data model without writing a single line of code. Open up XCode and create a new project using the Core Data Application template and call it MTCoreData.

If you open up the Models folder in your project, you will see that there is already a model file called MTCoreData_DataModel.xcdatamodel. Click it, and it should bring up the Core Data model editor (if not, click the Editor icon in your toolbar). There are four sections to the editor (figure 1).

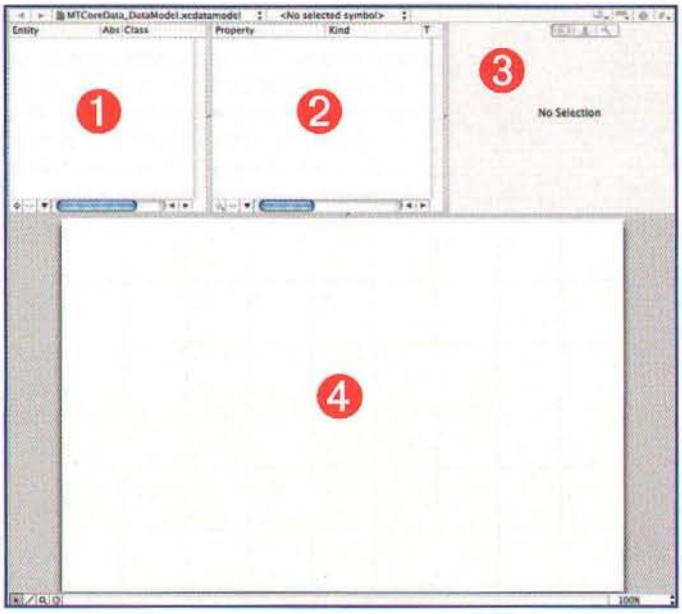


Figure 1. Blank Core Data Editor

This is the list of entities in your data model. The plus sign in the lower left, next to the scroll bar, will add an entity, which can be edited in section 3. The minus sign will allow you to delete entities, and the triangle next to the minus sign will allow you to change the view from a flat list to a hierarchical display.

This is the list of all the properties for the entity currently selected in section 1. The plus sign in the lower left allows you to add a new relationship, attribute, or fetched property. The minus sign will delete the currently selected property, and the triangle next to it will present a drop down that allows you to filter the property display based on type.

This is where the selected entity or property can be edited. If an entity is selected, but no property is, then this area will display the editable traits of the entity. If both an entity and a property are selected, then the editable traits of the property will be displayed here.

This is the object graph: a visual display of the data model. You can also edit many of the entity and property traits right in the object graph display if you want.

Go ahead and click on the plus sign in the lower left of section 1. This will add and select a new entity called **Entity**. The naming convention for entities is that they start with a capital, followed by lowercase letters except you capitalize the first letter of any new word (e.g. **FaxMachine**). Entities should be named as singular, not plural (not **FaxMachines**). Go ahead and change the name of the entity to **Book**. We don't need to call it **MTBook** because entity names only have to be unique within the data model, so there's no chance for a name conflict as there is with objective-C classes. Leave the class as **NSManagedObject**; we'll get into custom classes in a future article, but for now, leave it at the default value. We also won't be specifying a parent or making it abstract, so we are done with editing this entity.

Now, click on the plus sign at the lower left of the properties section and add an attribute called **isbn**. The naming convention for properties is the same as for objective-C instance variables: lowercase letters except where a new word starts (e.g. **faxMachine**). Unclick the **optional** checkbox – we want Core Data to require this field, but leave **transient** unchecked (transient properties do not get saved to the data store; this can be used for temporary or calculated properties, but are not commonly used without also creating a custom subclass of **NSManagedObject**).

Set the attribute's **type** to **String**. You'll notice that you get some new fields after changing the type. These allow you to do some validation without writing code. ISBN numbers must be ten characters, so we could set the minimum and maximum lengths both to 10, which would cause Core Data to reject input that is shorter or longer than ten characters. We won't do that, however, because there's another way to validate this input. We can also give the attribute a default value. We won't do that here, but we will for other attributes.

That leaves just one field: **Reg. Ex.**, which stands for "regular expression." If you're not familiar with regular expressions, it's well worth your time as a programmer to learn them; they often allow you to do a lot of work by writing a single, very compact string. They are also the key to using a number of the powerful command line tools that ship with OS X, such as grep, awk, and sed, not to mention the programming language Perl.

The reason we don't need to set a minimum or maximum length is because we can enforce that with a regular expression. ISBN values not only have to be 10 characters long, but the first nine characters must be numbers, and the final character must be a number or the letter X. Sure, we could write a half-dozen lines of code to enforce this, *if* we were creating a subclass of **NSManagedObject**, but the **Reg. Ex.** field gives us a way to do this without writing any code. We can make Core Data validate the entered value by simply typing the value **\d{9}[\dXx]** into the **Reg. Ex.** field. A regular expression tutorial is way beyond the scope of this article but, briefly, this string says that a valid input must have nine numeric digits followed by either another digit or an upper or lowercase x.

Add another string attribute called **title**. Give it a default value of **Untitled Book** and unclick **optional**. Also add string attributes called **publisher** and **comment**, both of which can be optional and have no default values. You can specify a minimum or maximum length if you want, but I've left them blank.

Core Data has no entity type corresponding to a URL, so we'll also use a string to store the book's URL. Create another string attribute called **url**, make it optional with no minimum or maximum length.

Next, we need to hold two dates: **dateReleased** and **dateRead**, so add two more attributes and give them the type of **Date**. Don't worry about giving a minimum, maximum or default value.

In **MTBook**, our variable **salesRank** was an **int**. In Core Data, there are a few different options for storing integers, the only difference among them is the amount of memory they use. Since Amazon sales ranks can be very big numbers, create a new entity called **salesRank** and use an **Integer 32** for the type. This can be optional, and we need no maximum or default value. We will put a minimum value of 0, however, since sales ranks have to be positive.

We also stored the cover image in **MTBook** as an **NSData**. Core Data does not have an image type, but it does have a type called **Binary Data** which can hold just about any type of data, much like **NSData**, so add another attribute called **coverImage** and assign it a type of **Binary Data**.

Well, that takes care of everything except authors. In the book information returned from Amazon, a single book can have multiple authors. We can represent this in Core Data by creating another entity and using relationships to link our two entities. Add a new entity called **Author**, again not abstract and with no parent entity. We only need to track one piece of information about an author—his or her name— so add a new entity to **Author** and call it **name**. Set the type to **String** and unclick **optional**. Next, we need to create a relationship between the **Author** entity we just created and the **Book** entity, which we can do by adding a relationship to **Author** and calling it **books**. Unclick optional and set the destination entity to **Book**. We will keep this as a "to-one" relationship, which will cause Core Data to create and store a new instance of an **Author** entity for every

author of every book. A more efficient way to do it would be to click the To-Many Relationship checkbox and reuse Author entities when an Author entity already exists, but that can't be done just using Cocoa Bindings and would require some code to enforce. For sake of brevity, implementing the more efficient version is left as an exercise for the reader.

Finally, we need to go back to the Book entity and define its relationship back to the Author entity we just created. Since we need to be able to store multiple authors in a single book, create a new relationship in the Book entity and call it authors. Leave it optional, since there are some books without a known author, and set the destination entity to Author. Set the inverse relationship to books, which tells Core Data that these two relationships are actually the same relationship viewed from different entities (as you do it, watch the object graph and you'll see the two lines connecting the Book and Author entities merge into a single line). This time, we do want to click the To-Many Relationship checkbox, since we need to be able to store more than one author for a book.

Save it. Your data model is done. We haven't written a single line of code, yet everything is now in place to initialize, use, persist, delete, and copy data.

Creating and Binding the Interface

This step, at first, works exactly as it has for years. Double-click MainMenu.nib in XCode, which will open up Interface Builder where you can design your interface. I made mine look like this:

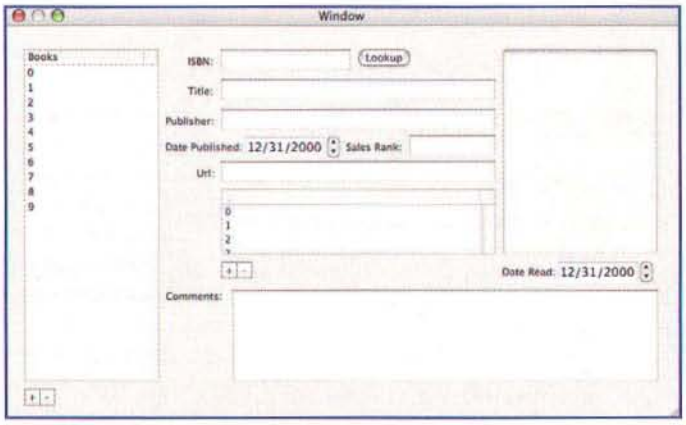


Figure 2. The window layout

In order to implement the Lookup button, we will have to write some code, but surprisingly little. Before we do that, however, we'll use Cocoa Bindings to link the user interface directly to our Core Data context, which will let us add, edit, and delete objects without writing code.

In order to proceed, we do need to add a couple of non-visual items to our nib. These are NSArrayController instances. NSArrayControllers are a handy part of Cocoa Bindings that take care of most of the potential interactions a user or user interface element could have with an array of data. With the release of Core Data, NSArrayControllers can be configured to automatically load their data from the context. We need one controller to manage all the books that have been entered, and one to manage the authors of the currently selected book.

To add an NSArrayController instance to your nib, simply drag the icon that looks like three little green boxes from the Controllers palette (figure 3) to your nib file's main window and name it Books.

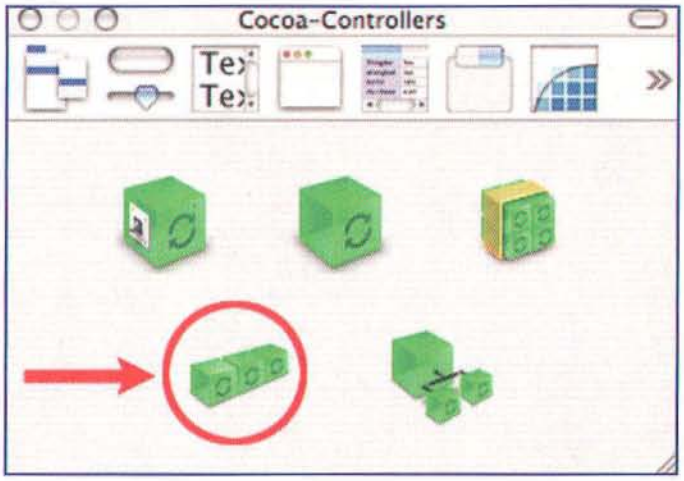


Figure 3. NSArrayController icon

Single-click the new array controller, and press ?1 to bring up the array controller's attributes in the inspector palette. Make sure that the mode is set to Entity and Automatically Prepares Content is selected. Type Book into the Entity Name field. This is how Cocoa Bindings knows to populate the array controller from Core Data with all the Book entities.

We also have to tell Cocoa Bindings the appropriate context from which to populate the array controller. We can do this by pressing ?4 and bringing up the array controller's bindings in the inspector. The bottommost entry on the bindings palette, under the Parameters section, is managedObjectContext. Click this to expand it, and click the Bind checkbox. For the Bind To field select MTCoreDataAppDelegate, the application delegate that was automatically created for you.

In the Model Key Path field, type in managedObjectContext, which is the name of an accessor method created automatically for you in MTCoreDataAppDelegate that returns the application's context. Leave the Value Transformer field blank; we'll talk about value transformers in a future article.

That takes care of the NSArrayController that will be used for the books table on the left side of the window. We also need one for the authors table. This one's a tad bit trickier because, unlike the Books array controller, we only want to show those authors that correspond to the currently selected book. Drag another NSArrayController to your nib and rename it Authors.

In the controller's attributes, make sure it's set to Entity with a name of Author and that Automatically Prepares Content is selected. In the array controller's bindings, set the managedObjectContext exactly as you did with the Books array controller.

Now, we'll tell it to only load those authors who are authors of the currently selected book. We do this with the contentSet binding under the Controller Content heading. Click on it to expand it, and click the bind checkbox. Bind this to the Books array controller with a Controller Key value of selection, which should be the default and tells Core Data to pull data from the selected item in the Books array controller. In the Model Key Path field, type authors, which is the name of the relationship in our Book entity that we want to use for populating this array controller.

Now, let's bind up the user interface elements. Double-click the books table down the left side, then double-click again so that the table column, and not the table or scroll view, is selected. Type ?4 to bring up the bindings and click on the value binding to expand it. Bind it to the Books array controller with a Controller Key of arrangedObjects, which binds the column to the objects in the Books array, in the correct sort order if one is specified. Finally, in the Model Key Path field, type in title, which is the name of the attribute we want to display in this column. The only other thing we need to do is to check Continuously Updates Value, which ensures that the data model and user interface are always kept synchronized, not just when you tab out of a field.

Next, double-click the other table on your interface—the authors list—twice, so that the table column is selected and bring up the bindings for it. This time, bind the value parameter to the arrangedObjects key of the Authors array controller and type in name for the key path, which tells it to display the name of the author in this column.

That takes care of the two tables on our interface. The individual fields are just as easy. Since we want to display data about the book currently selected in the left-hand table, we need to bind each field to the selection key of the Books array controller. For every field, Continuously Updates Values should be checked. The only difference in the bindings of the different fields is the Model Key Path field, which should be the attribute name (from the Book entity) that you want displayed in that field. You can also check Validates Immediately if you want the input value to be checked when tabbing out of the field rather than at save time. Here's what the value binding should look like for the isbn field:

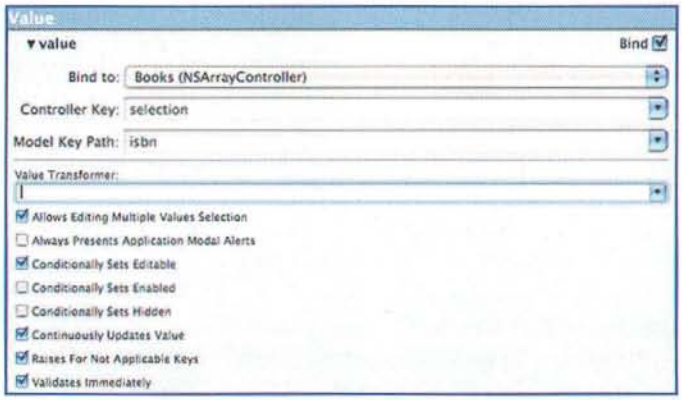


Figure 4. Bindings for ISBN field

Once you've bound all the other fields on the interface (make sure the NSImageView is set to enabled), there's just one task remaining before we try it out. See those little buttons with the plus and minus signs? We've already created the ability to edit data in the context. Now, we're going to let these buttons add and delete Book and Author entities to the context, again without writing any code. Control-click the plus button in the lower left of the window and drag to the Books array controller. Connect the button's target to the add: outlet. Repeat with the minus button next to it, but connect the target instead to the remove: outlet. Do the same thing for the two buttons below the authors table, but bind to the Authors array controller instead of the Books array controller.

Let 'Er Rip...

Well, we haven't written any code, but go ahead and compile and run the application. Press the plus button and you'll get a new, untitled book. Edit any of the fields or drop a picture onto the image view, and then quit. When you launch the program again, your data is still there. If a book is selected, and you press the minus button, the book gets deleted. Hit ?Z and it comes back. Look Mom: free undo! Change the title of the book, and the table column on the left instantly reflects the change. Pretty easy, huh?

A Little Spit and Polish

We can do a few quick things to add some professional touches to our application. We can link the Save menu item to the saveAction: outlet of MTCoreDataAppDelegate, and allow the user to save any time in the fashion they're used to, rather than just at when the application quits. Want to give the ability to revert? Still easy, but we do have to add bit of code for that.

Listing 1: MTCoreDataAppDelegate.m

revertAction

Adding these two methods to our application controller and then binding the Revert menu item to the first one allows the user to revert to the last saved version. The first method simply invokes a sheet asking if they really want to revert. The second method is called when the sheet ends, and does a rollback if the user confirmed. Rollback is the SQL term for reverting: The EOF heritage shows here in the choice of method name.

```
- (IBAction)revertAction:(id)sender
{
  NSBeginAlertSheet(@"Are you sure?",@"No",@"Yes",nil,
  window,self,nil,
  @selector(reverSheetDidEnd:returnCode:contextInfo:),
  nil,@"Are you sure you want to revert?");
}
- (void)reverSheetDidEnd:(NSWindow *)sheet
  returnCode:(int)returnCode
  contextInfo:(void *)contextInfo
{
  if (returnCode == NSAlertAlternateReturn)
    [[self managedObjectContext] rollback];
}
```

Another quick bit of polish we can do is to make the minus buttons enabled only when a book is highlighted and, therefore, able to be deleted. We can do this by binding its enabled binding

parameter to the canRemove controller key of the Books array controller. Likewise with the authors remove button to the Authors array controller.

Finally, one last nicety we'll handle this month is to tell the Books array controller to sort based on the title of the book. This will cause the table on the left hand side to display the titles in alphabetical order, since it's bound to the arrangedObjects outlet of that NSArrayController. Unfortunately, this one takes a bit of code, but not much.

Listing 2: MTCoreDataAppDelegate.m

Sorting the Books array controller

To sort our books table, we need to create a sort descriptor and supply it to our array controller. Create an IBOutlet instance variable in the MTCoreDataAppDelegate.h header file. Make it an NSArrayController and give it a name of books. In interface builder, bind the Books array controller to this outlet (you may have to drag MTCoreDataAppDelegate.h over to Interface Builder to make sure it knows about your changes). The best place to specify our sort descriptor is when our application is finished launching, so we'll use the handy notification method applicationDidFinishLaunching: that we get automatically by virtue of being the NSApplication delegate.

```
- (void)applicationDidFinishLaunching:
  (NSNotification *)aNotification
{
  // This sort descriptor says to sort alphabetically, in ascending order (A
then B then C)
  // based on the property called title.
  NSSortDescriptor* sort = [[NSSortDescriptor alloc]
    initWithKey:@"title" ascending:YES];
  // Tell our array controller to use our sort descriptor
  [books setSortDescriptors:
    [NSArray arrayWithObject: sort]];
  [sort release];
}
// The Books array controller also has to be told to re-sort when something
that affects
// the sort order changes. In our case, the title field is the only one that can
affect the
// sort order of the book table, so we'll bind it to this method in Interface
Builder.
// Doing so tells the array controller to re-sort based on the sort descriptor
we gave it
// any time the title value is changed on the interface earlier
- (IBAction)rearrangeBooks:(id)sender
{
    [books rearrangeObjects];
}
```

One Last Thing

MTCoreDataAppDelegate, the application delegate that was created automatically for you has an accessor method, also created automatically, that it's worth taking a look at because it's the key to changing the Core Data storage options. By default, Core Data uses an XML file. You can change this to use a binary file or an embedded SQLite database, by making a minor tweak to the managedObjectContext method. Look at the items in bold in the following excerpt from mangedObjectContext; the first is the filename that will be used for the persistence store, the second tells Core Data how to store the data.

Listing 3: MTCoreDataAppDelegate.m

Changing Core Data Storage Options

```
url = [NSURL fileURLWithPath: [applicationSupportFolder
  stringByAppendingPathComponent: @"MTCoreData.xml"]];
coordinator = [[NSPersistentStoreCoordinator alloc]
  initWithManagedObjectModel: [self managedObjectModel]];
if ([coordinator addPersistentStoreWithType:NSXMLStoreType
  configuration:nil URL:url options:nil error:&error])
```

By changing the value NSXMLStoreType to one of the other available options, you change how Core Data will persist your data. Changing the value to NSBinaryStoreType will cause Core Data to store the data as a binary file. Storing it in NSSQLiteStoreType will cause Core Data to create an embedded SQLite database. There is actually a fourth possible option here, but it's one that doesn't make much sense to use: NSInMemoryStoreType. Specifying the last option will cause Core Data to discard all your data at quit time, so won't generally be recommended by people who like you.

Although not strictly required, it is good form to change the filename used to create url so that a different file extension is used if you change the store type; having a binary or SQLite file with a .xml ending is likely to cause confusion. Feel free to use any extension that seems appropriate for your situation. Personally, I use .sqlite3 when using NSSQLiteStoreType and no extension when using NSBinaryStoreType, but use whatever tickles your fancy.

Conclusion

At this point, we've written around thirty lines of code—and that's even counting lines with just a bracket—and we have a full-fledged application capable of saving text, dates, and images and displaying them in an ordered table view. Our application validates field values, allows undo and revert and generally gives the user a good chunk of the functionality they expect from a finished application. In our next article, we'll re-implement the Amazon ISBN lookup to show how we add, edit, and delete data programmatically, then we'll dive into some more advanced functionality requiring us to subclass NSManagedObject.

This article should have given you enough of a glimpse of Core Data's possibilities to understand why many developers are salivating over it. Core Data can give an incredible boost to your productivity and allow you to create more robust, more easily maintained applications faster than previously possible and, frankly, it's hard to hate that.

About The Author

Jeff LaMarche is a long time Mac user and programmer who still hasn't figured out what he wants to do when he grows up, which is okay, because he probably won't. You can reach him at jeff_lamarche@mac.com.

TIPS FROM

by Aaron Hillegass, Chris Campbell, Marquis Logan

Style For Cocoa Programmers

As developers and teachers, we spend a lot of time dealing with issues of style. Not just because we like to look cool for the ladies; code that is written by a stylish programmer is easier to read and maintain. The Cocoa community has developed idioms, and if you want other developers to be able to easily comprehend your code, you should follow those idioms.

Issue #1: Whitespace

Here is a perfect setter method that explicitly invokes key-value observing:

```
- (void)setBorderColor:(Color *)c
{
    if (c == borderColor) {
        return;
    }
    [self willChangeValueForKey:@"borderColor"];
    [c retain];
    [borderColor release];
    borderColor = c;
    [self didChangeValueForKey:@"borderColor"];
}
```

Notice the use of whitespace. In particular, we do not put spaces after colons. We do not put spaces after the type of the parameter (That is, "(Dog *)").

Also, notice that there are no comments. What this method does is self-evident from its name. Too many comments can be just as annoying as too few.

Issue #2: Grouping Methods

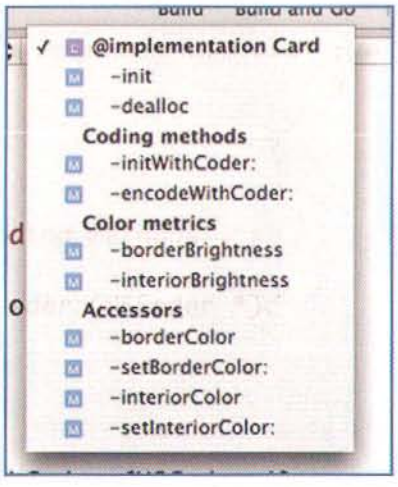
In an implementation file, methods should be grouped in a meaningful manner. In particular, we try to follow this order:

Class methods
Initializers and dealloc
`copyWithZone:`
`isEqual:`
`hash`
`initWithCoder:` and `encodeWithCoder:`
action methods
other methods (subgroup meaningfully)
delegate methods
accessor methods (in setter/getter pairs)

Use #pragma mark to show where a group of methods begins. The information will show up in the popup in Xcode. For example, you might insert these lines:

```
#pragma mark Coding methods
#pragma mark Color metrics
#pragma mark Accessors
```

So that your Xcode popup would look like this:



Issue #3: Naming methods

If you have a method that returns the brightness of a border, it should be named thusly:

```
- (float)borderBrightness;
```

We do not put articles on method names:

```
- (float)theBorderBrightness;
```

We do not prefix methods with get-:

```
- (float)getBorderBrightness;
```

The exception to this rule is if the method is pass-by-reference:

```
float r, g, b;
[borderColor getRed:&r
                blue:&b
               green:&g
               alpha:NULL];
```

If a method takes one argument and is prefixed with set-, it should be an accessor. In particular, an action method should never be prefixed with set-.

Issue #4: What Goes In The .h File

The primary purpose of your .h file is so that I can create an instance of your class and send messages to it. Many people put too much information in their .h files. Don't declare any of the following in the .h file:

1) Delegate methods.
2) Private methods.
3) Any methods in a protocol that you conform to. For example, if you have:

```
@interface Chowder : NSObject <NSCopying>
```

There is no reason to declare copyWithZone: in your .h file.
4) Any inherited or overridden methods.

Issue #5: Declaring Local Variables

Declare temporary variable in a "just-in-time" manner. When I'm reading your code, I don't want to have to scroll to the beginning of the method to figure out the type of a local variable. Move the declaration as close to the first use as possible.

The Analysis and Design Workshop

In a short column, it is easy to discuss small issues like code formatting and variable naming. For larger and deeper issues, we need a much bigger forum. At the ranch in August, we are hosting an "Analysis and Design Workshop." John Graziano, who had a major part in the design of Xcode, WebObjects Builder, and many of the animation systems at Pixar, will lecture and guide hands-on exercises that teach good design practices. He is going to focus on creating designs that make change and maintenance easy. John started developing these ideas at NeXT, and they have received much acclaim within NeXT, Apple, and Pixar. Bring your specifications, and go home with the beginnings of a really good design for your app. The workshop is August 15 - 19 in Atlanta. We hope to see you there.

MU

DEV DEPOT

DevDepot has it all!

Get More out of your Mac!

DevDepot sells the tools, toys and technology to put more muscle into your Mac. Visit our online store today for special offers and great new products. **www.devdepot.com**

Television is about to get a whole lot more interesting.

NOW ONLY $324.99

EyeTV 200

The future of Television

Watch TV on your Mac!

Pause live tv! Record, edit and archive!

go **digital**

Convert your analog video tapes to digital in realtime, then burn them to DVD! *

Watch TV on your Mac

EyeTV features a 124-channel cable-ready analog tuner and DVD quality MPEG-2 video encoder.

Record TV on your Mac and Archive to Disk

Collect and organize your favorite shows to **watch whenever you want**, then update video content to DVD.* (Toast 6 Titanium required).

Don't miss a thing

Use EyeTV's Electronic Program Guide to find exactly what you are looking for, and program EyeTV to record it. **Program EyeTV from anywhere via the Internet.**

NEW!

New Export Functionality

EyeTV now lets you export to iMovie®, iDVD® and DVD Studio Pro®, making it easier to create professional quality recordings.

from elgato

Features and Benefits

- Record TV otn Your Mac
- Edit Out Unwanted Content
- Archive Recorded TV To DVD*
- MPEG-2 Video Encoding
- Digitize Analog Video
- The Speed And Power Of **Firewire**

Save Our Screens - 102 - How To Write A Mac OS X Screen Saver, Part 2.

By David Hill

Introduction

In the last article, we covered some introductory material and put together a basic screen saver. We also covered some debugging tips that should have helped you get your own projects started. Now that you've got a simple screen saver up and running and you've got a grasp of the ScreenSaverView class, let's take things a bit further and add a simple configuration sheet. We'll need to make several modifications to our project.

1. Change the code in projectNameView.h to this:

```
#import <ScreenSaver/ScreenSaver.h>
#define kConfigSheetNIB @"ConfigureSheet"

@interface projectNameView : ScreenSaverView {
    IBOutlet NSWindow* configureSheet;
    IBOutlet id shouldRotateCheckbox;
    BOOL isRotatingRectangles;
    BOOL mDrawBackground;
}
- (IBAction) cancelSheetAction: (id) sender;
- (IBAction) okSheetAction: (id) sender;
@end
```

2. Change the hasConfigureSheet method to return YES.
3. Change the code in the configureSheet method to this:

```
if ( configureSheet == nil ) {
    [NSBundle loadNibNamed: kConfigSheetNIB owner:
self];
  }
  [shouldRotateCheckbox setState:
isRotatingRectangles];
  return configureSheet;
```

4. Conditionalize the rotation code in drawRect: like so:

```
  if ( isRotatingRectangles ) {
    NSAffineTransform* rotation =
            [NSAffineTransform transform];
    float degrees = SSRandomFloatBetween( 0.0, 360.0
);
    [rotation rotateByDegrees: degrees];
    [rotation concat];
  }
```

5. Add the following line to initWithFrame:isPreview: before returning self:

```
isRotatingRectangles = YES;
```

6. Add a cancelSheetAction: method that should look like:

```
  - (IBAction) cancelSheetAction: (id) sender {
    //  close the sheet without saving the settings
    [NSApp endSheet: configureSheet];
  }
```

7. Add an okSheetAction: method that should look like:

```
  - (IBAction) okSheetAction: (id) sender {
    //  record the settings in the configuration
sheet
    isRotatingRectangles = [shouldRotateCheckbox
state];
    [NSApp endSheet: configureSheet];
  }
```

That's the easy part to explain since it only involves code. We've got all the code in place for our configuration sheet, but where's the sheet itself? That would be the tricky part. Fire up Interface Builder and create a new empty NIB. In order to connect our screen saver view class to elements in the NIB and vice versa, we're going to need to teach IB about our custom class. By default, IB doesn't know about the ScreenSaverView superclass so we'll start there. Arrange your project window and the NIB window such that you can see both. Open the ScreenSaverView.h header file in Xcode and drag the header (either the window title bar proxy or the header icon in the Groups and Files outline) to the NIB window. You should see a green cursor with a plus in it when the cursor is over the IB window as seen in Figure 1. That's a good sign and it means that IB will parse the file and absorb the ScreenSaverView class information when you drop the header file into the IB window.

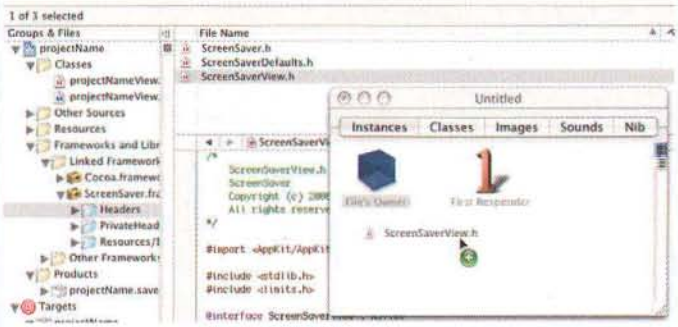


Figure 1: Teaching Interface Builder about ScreenSaverView

Now we're ready to teach IB about our custom projectNameView class but there's a simpler way than dragging header files around and juggling windows. Switch to IB, switch to the Classes tab in the NIB window, and select Read Files... from the Classes menu. In the resulting dialog, navigate to your project folder, select your projectNameView.h header, and click Parse. As a final step, we need to tell IB that our NIB will be owned (this usually means loaded) by our custom class. Switch the NIB file view to Instances and select the File's Owner. Open the Info window (SHIFT-CMD-I) and select Custom Class (CMD-5) from the popup menu. Find projectNameView in the list and select it to change the class.

At this point, IB knows about our custom header file and we can start creating the sheet and wiring it up. The screen saver engine is expecting the configureSheet method to return an NSWindow so let's give it one. Show the palette in IB and select the Windows button from the toolbar. Drag a normal window into the NIB document window to add it to your NIB. Switch to the Controls portion of the IB palette and add a switch (a checkbox for you Carbon people) and two buttons to your window. Change one button to say OK and the other to say Cancel. Change the switch to say something like Rotate the rectangles but the exact title doesn't matter. The window should look something like the one in Figure 2.

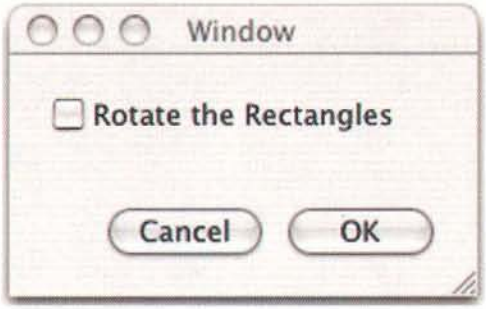


Figure 2: Our finished configuration window

With our controls in place, we need to make our final connections. The view needs to know about the sheet in order to close it and the controls need to know how to tell the view they've been clicked. Control-drag from the File's Owner to the window (either in the Instances view or the titlebar of the window itself) and set the configureSheet connection. See Figure 3 if you need help. Control-drag from File's Owner to the checkbox and connect it to the shouldRotateCheckbox outlet. Control-drag from the OK and Cancel buttons to File's Owner and set the okSheetAction: and cancelSheetAction: connections, respectively. Figure 4 shows the okSheetAction: connection.

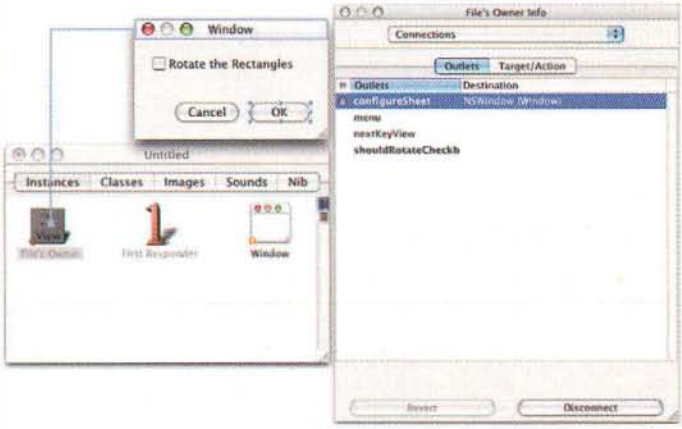


Figure 3: Connecting the configureSheet outlet to our window

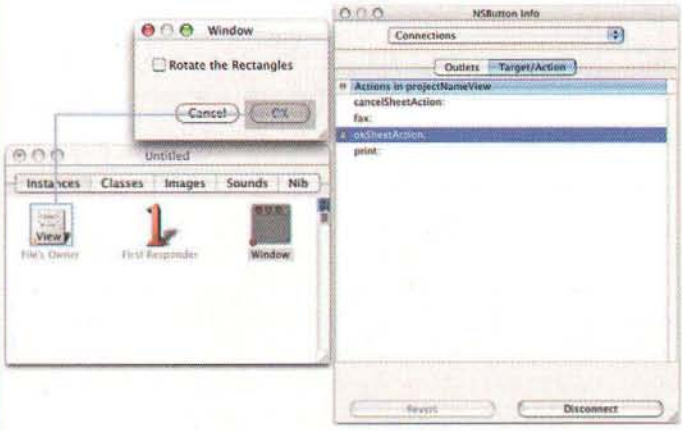


Figure 4: Connecting the OK button to okSheetAction:

Now we're getting somewhere. Save this new NIB file as ConfigureSheet into your project directory and IB will ask if you'd like to add it to your currently open Xcode project. Click Add. Build your screen saver and try it out. If we're lucky, the saver will still build and work. Open the Desktop and Screen Saver preferences pane, select your saver, and click the Options button to see your new configuration sheet. Hopefully it looks a lot like Figure 5. Turn the rotation checkbox off then close the sheet and watch what happens to the Preview. Looks good, right? Ah, but there's one small problem. Click the Test button and see for yourself.

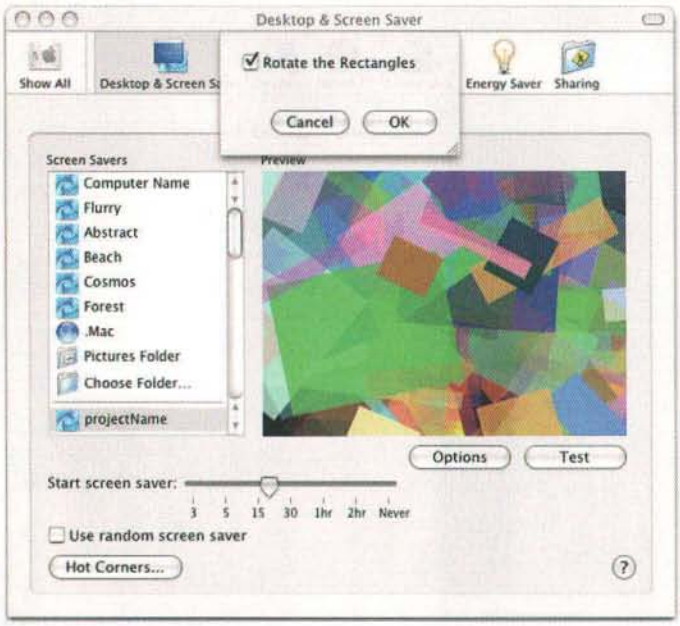


Figure 5: The configuration sheet in action

Why is the screen saver still rotating rectangles when we clearly turned off the checkbox in the configuration sheet? The answer is simple and this is a good excuse to drive home the underlying reason. The screen saver engine creates a separate instance of the current saver whenever it needs to blank a different area. The Preview pane gets an instance. The Test run of the saver gets a new, separate instance, and if you run the saver for real via a hot corner you'll get yet another instance. Those of you with multiple monitors may have already noticed that the saver draws different things on each monitor which shows that each monitor also gets its own instance of the screen saver. For the most part, this is a good thing but there are times when you'd like to share some state between the different instances. User preferences are the most common case and the ScreenSaver framework provides a convenient solution in the form of the ScreenSaverDefaults class and its defaultsForModuleWithName: method.

In the next section, we'll use defaultsForModuleWithName: to store and retrieve the current isRotatingRectangles setting so that each screen saver instance will do the right thing.

Defaults

The NSUserDefaults system provides a very handy way of storing and retrieving user preferences in such a way that applications (and screen savers) don't have to care where the defaults are stored. Your code can just make the appropriate method invocations and the frameworks handle the rest. For screen savers, the ScreenSaverDefaults class makes things even easier. It provides a single method, defaults ForModuleWithName:, in addition to the standard methods provided by NSUserDefaults. Note that it is extremely important that your screen saver use the ScreenSaverDefaults mechanism rather than NSUserDefaults, which is keyed off the application that is reading and writing defaults. Since screen savers get loaded in by more than one application, you'll need to use the defaultsForModuleWithName: method of ScreenSaverDefaults to keep your preferences straight. As you might expect from the name, a screen saver uses defaultsForModuleWithName: to obtain a ScreenSaverDefaults reference that it can then use to load and store preferences. Let's see how that works in practice by adding code to handle our one preference value, isRotatingRectangles. First, we need to make sure that the defaults system knows what our global default for isRotatingRectangles should be (would that be a default default?) by registering a default dictionary.

1. Add the following define statements to projectNameView.h:

```
#define kDefaultsModuleName
            @"projectName_Random_Rectangles"
  #define kDefaultsIsRotatingRectanglesKey
            @"isRotatingRectanglesDefault"
  #define kDefaultsYesString        @"YES"
```

2. Declare a new helper method in projectNameView.h:

```
- (ScreenSaverDefaults*) defaults;
```

3. Add the new helper method definition to projectNameView.m:

```
- (ScreenSaverDefaults*) defaults {
    // there's no need to cache this value so we'll make a handy accessor
    return [ScreenSaverDefaults defaultsForModuleWithName:
            kDefaultsModuleName];
}
```

4. Replace the assignment to isRotatingRectangles in initWithFrame:isPreview: with:

```
// find the defaults we should use for this screen saver
  ScreenSaverDefaults* screenSaverDefaults =
            [self defaults];
  // create a dictionary to contain our global default values
  NSDictionary* defaultDict = [NSDictionary
            dictionaryWithObject: kDefaultsYesString
            forKey: kDefaultsIsRotatingRectanglesKey];
  // register those global defaults
  [screenSaverDefaults registerDefaults: defaultDict];
  // now we can read in the current default value knowing
  // that we'll always get the user defaults or our global defaults
  // if no user defaults have been set
  isRotatingRectangles = [screenSaverDefaults
            boolForKey: kDefaultsIsRotatingRectanglesKey];
```

Before we move on, I should make a few comments about this code. Note that the defaultDictionary is created in an autoreleased state so we don't have to worry about cleaning it up to prevent leaks. As you'll see later, we also don't have to read the defaults back in right away since we'll be checking the default value of isRotatingRectangles before drawing each frame. So why is the boolForKey: call here? I included it for several reasons, not the least of which was reinforcement of the defaults idiom. It also makes sure that we've got the correct value of isRotatingRectangles from the very start in case we forget to check it later in the code. For example, the configuration sheet code sets the value of the checkbox based on the current value of isRotatingRectangles. If we forget to load the appropriate value before bringing up the sheet, the checkbox will be out of sync with the current default value and users will get confused. As a final comment on the initWithFrame:isPreview: code, note that we've now got two string key values: kDefaultsModuleName that represents the name of the module in the defaults system and kDefaultsIsRotatingRectanglesKey that we'll need to use whenever we want to access our default value. What else needs to change in order to correctly handle the default value? Well, as I mentioned above we'll be checking the default value in the drawRect: method so that we know whether or not to rotate the rectangles.

5. Add this code before if (isRotatingRectangles) in the drawRect: method of projectNameView.m:

```
isRotatingRectangles = [[self defaults]
            boolForKey: kDefaultsIsRotatingRectanglesKey];
```

Checking here makes sure that if anybody changes the default value, even while we're running, we'll get the correct value and change our drawing style in mid-stream. One other important concept for user defaults that can be shared between different instances is that you need to update the defaults whenever the user changes a value. In our sample, the only place that the user gets to change anything is in the configuration sheet so we'll need to add a bit of code there.

6. Add the following code to the okSheetAction: method before we close the sheet in the projectNameView.m file:

```
// write out the current default so other instances of the saver pick up the change, too.
    [[self defaults] setBool: isRotatingRectangles
            forKey: kDefaultsIsRotatingRectanglesKey];
// update the disk so that the screen saver engine will pick up the correct values
    [[self defaults] synchronize];
```

This code takes the latest version of the isRotatingRectangles variable and stores it in the screen saver's defaults. At the end there is a call to synchronize, but why? The synchronize call is there so that the true screen saver mode (invoked via the hot corner or system idle state) picks up the correct default value even if the System Preferences application is still running. This is technically just an implementation detail, but it turns out that while the Preview and Test modes share the in-memory defaults, due to the fact that they're both instantiated from the System Preferences process, there is a separate process that handles the full invocation of the screen saver module when your system has been idle for too long or you move the mouse into the Activate Screen Saver hot corner. The screen saver instance loaded by this ScreenSaverEngine application can only retrieve the latest defaults that have been written to disk and for screen saver modules that only seems to happen at two times: when the user quits the System Preferences application and when the screen saver explicitly calls synchronize.

For the purposes of this article, I've used a very generic scheme for the configuration sheet and its defaults. In this scheme the defaults are only updated and synchronized once the user is done changing all of the values. No changes are made to any defaults or screen saver state variables while the options sheet is open and the defaults only get updated once per invocation of the configuration sheet rather than on each control change. This style of configuration sheet also gives the user the option of canceling any changes they've made to the settings. The configuration sheet is one way to interact with and control a screen saver but screen savers can also react to a set of user input events much like any normal application. You can use these events to enable and disable effects, toggle status displays, and even turn your screen saver into a game! (Anybody remember Lunatic Fringe?) In the next section, we'll take a look at the methods you'll need to override to catch these events and handle them in your screen saver.

Handling Events

The view system in Cocoa provides a rich set of user input events that we can tap into while a screen saver is running. The implementation in the ScreenSaverView superclass uses most of the events as a signal to wake up the screen saver so the first thing we need to do is override that behavior in our subclass to prevent the saver from waking prematurely. Note: don't override all of the events or you won't be able to wake the screen saver at all!

1. Override the key handling code by adding the following implementations of methods inherited from the NSResponder class to projectView.m:

```
- (void)keyDown:(NSEvent *)theEvent {
    // handle any necessary keyDown events here and pass the rest on to the superclass
    [super keyDown: theEvent];
}
- (void)keyUp:(NSEvent *)theEvent {
    // handle any necessary keyUp events here and pass the rest on to the superclass
    [super keyUp: theEvent];
}
```

These methods simply pass all keyDown and keyUp events up to the superclass for it to handle. If, for example, we don't want keyDown events to wake the screen saver we can change the implementation of keyDown: by commenting out the call to

super. However, we'll want to handle some keys ourselves and let the superclass handle others.

1. In that case, we can change the keyDown: code to look more like this:

```
// handle any necessary keyDown events here and pass the rest on to the superclass
  if ( [[theEvent charactersIgnoringModifiers]
          isEqualTo: @"g"] ) {
    // the user pressed the "g" key so draw the next 50 rectangles in grayscale
    grayscaleRectanglesLeft += 50;
    // note that the "g" key will no longer wake up the screen saver
  } else {
    [super keyDown: theEvent];
  }
```

We'll also need to add code to declare grayscaleRectanglesLeft and check for its value while drawing so make the following additional changes.

2. Add the declaration for grayscaleRectanglesLeft to projectNameView.h:

```
int grayscaleRectanglesLeft;
```

3. Give grayscaleRectanglesLeft an initial value in initWithFrame:isPreview:

```
grayscaleRectanglesLeft = 0;
```

4. Change the color setup in drawRect: from this:

```
float red = SSRandomFloatBetween( 0.0, 1.0 );
    float green = SSRandomFloatBetween( 0.0, 1.0 );
    float blue = SSRandomFloatBetween( 0.0, 1.0 );
    float alpha = SSRandomFloatBetween( 0.0, 1.0 );
    [[NSColor colorWithDeviceRed: red
            green: green blue: blue alpha: alpha] set];
```

to this:

```
  if ( grayscaleRectanglesLeft ) {
      float white = SSRandomFloatBetween( 0.0, 1.0 );
      float alpha = SSRandomFloatBetween( 0.0, 1.0 );

      [[NSColor colorWithDeviceWhite: white
            alpha: alpha] set];
      grayscaleRectanglesLeft--;
    } else {
      float red = SSRandomFloatBetween( 0.0, 1.0 );
      float green = SSRandomFloatBetween( 0.0, 1.0 );
      float blue = SSRandomFloatBetween( 0.0, 1.0 );
      float alpha = SSRandomFloatBetween( 0.0, 1.0 );
      [[NSColor colorWithDeviceRed: red
            green: green blue: blue alpha: alpha] set];
    }
```

If you run the new version of the screen saver, you'll find that while all the other keys still wake up the saver, the "g" key causes the saver to draw a series of rectangles in shades of gray instead of the usual random colors. You can further modify keyDown: to intercept and handle other keys to do whatever you like but there are some useful keys, most notably the function and arrow keys, that don't have simple character equivalents. The trick is knowing how to interpret the result of charactersIgnoringModifiers for these keys. The short answer is, you don't. The longer answer is that you'll need to extract the unichar value of the character before comparing it to one of the constants like NSUpArrowFunctionKey. Change the implementation of keyDown: again to look like the following code to see how to handle arrow keys as well.

1. Change keyDown: to look like this:

```
  NSString* eventCharacters =
      [theEvent charactersIgnoringModifiers];
  unichar firstCharacter =
      [eventCharacters characterAtIndex: 0];
  // handle any necessary keyDown events here and pass the rest on to the superclass
  if ( [eventCharacters isEqualTo: @"g"] ) {
    // the user pressed the "g" key so draw the next 50 rectangles in grayscale
    grayscaleRectanglesLeft += 50;
  } else if ( firstCharacter == NSUpArrowFunctionKey ) {
    // the user pressed the up arrow so make the rectangles taller
    rectangleHeightMultiplier *= 2;
  } else if ( firstCharacter == NSDownArrowFunctionKey ) {
    // the user pressed the down arrow so make the rectangles shorter
    rectangleHeightMultiplier /= 2;
  } else if ( firstCharacter == NSRightArrowFunctionKey ) {
    // the user pressed the right arrow so make the rectangles wider
    rectangleWidthMultiplier *= 2;
  } else if ( firstCharacter == NSLeftArrowFunctionKey ) {
    // the user pressed the left arrow so make the rectangles narrower
    rectangleWidthMultiplier /= 2;
  } else {
    [super keyDown: theEvent];
  }
```

2. Add the declarations to projectNameView.h

```
  float rectangleHeightMultiplier;
  float rectangleWidthMultiplier;
```

3. Add some initial values to initWithFrame:isPreview: in projectNameView.m

```
// set the starting multipliers to 1.0
  rectangleHeightMultiplier = 1.0;
  rectangleWidthMultiplier = 1.0;
```

4. Change the rectToFill line in drawRect: to this:

```
NSRect rectToFill =
      NSMakeRect( startingX, startingY,
          width * rectangleWidthMultiplier,
          height * rectangleHeightMultiplier );
```

Note that we're still handling "g" and passing other non-arrow events on to the superclass. With these changes, the arrow keys now control how wide and tall (or narrow and short) the random rectangles tend to be. Your screen saver might also need to handle the modifier key event which covers SHIFT, COMMAND, OPTION, and CONTROL. Unlike the other keys, the modifier keys don't send a keyDown: message. Instead, your screen saver needs to override the flagsChanged: method in your projectNameView.m to catch modifier key events.

1. Add the following method to projectNameView.m

```
- (void)flagsChanged:(NSEvent *)theEvent {
  // toggle the current sense of the isRotatingRectangles flag
  // while the user is holding down the OPTION key
  if ( [theEvent modifierFlags] & NSAlternateKeyMask ) {
    reverseSenseOfIsRotatingRectangles = YES;
  } else {
```

```
        reverseSenseOfIsRotatingRectangles = NO;
    }
}
```

2. Add the following declaration to the projectNameView.h file:

```
BOOL reverseSenseOfIsRotatingRectangles;
```

3. Set the initial value in initWithFrame:isPreview:

```
// start out with isRotatingRectangles meaning what it says
reverseSenseOfIsRotatingRectangles = NO;
```

4. Check the value in drawRect: by replacing the rotate code with:

```
BOOL rotateThisRectangle = isRotatingRectangles;
if ( reverseSenseOfIsRotatingRectangles ) {
    rotateThisRectangle = !rotateThisRectangle;
}
if ( rotateThisRectangle ) {
    NSAffineTransform* rotation =
        [NSAffineTransform transform];
    float degrees = SSRandomFloatBetween( 0.0, 360.0 );
    [rotation rotateByDegrees: degrees];
    [rotation concat];
}
```

With those small changes, your screen saver should now temporarily reverse the sense of the isRotatingRectangles flag as long as you hold down the OPTION key. The only other events that you're likely to want to capture are mouse events and they're just as easy. NSResponder provides a series of methods you can override for mouse movement, dragging, clicks, and even scroll wheel activity. Depending on your requirements, you might not need to trap and deal with every event right when it happens. If your screen saver merely adjusts its drawing based on the current mouse position, you may be able to simply query the system inside drawRect: using the NSEvent class method mouseLocation which returns the current mouse position in global coordinates.

Note: watch out for multiple monitors and Preview mode here. Don't assume that the mouse position will remain within the bounds of your view or even the main display. Users with multiple monitors will be disappointed if your screen saver fails or misbehaves just because they've plugged in another display.

Different Drawing APIs

Up until this point, we've been using the drawing functionality provided by Quartz 2D and built into Cocoa. While NSBezierPath does provide for easy, accessible drawing code, there are times when you need to do more that draw lines, rectangles, curves, etc. This is often the case if you already possess some drawing code that you're merely trying to wrap up in a screen saver. The first step beyond NSBezierPath is to use NSImage to load and draw images during drawRect:, perhaps to provide a backdrop or for use as sprites in your animation. NSImage is very simple to use and provides support for many common image types. In a typical screen saver, you'll store your images in the Resources folder within the screen saver bundle and load them in with code similar to the following:

```
NSBundle* saverBundle =
    [NSBundle bundleForClass: [self class]];
NSString* imagePath = [saverBundle
    pathForResource: @"test image" ofType: @"JPG"];
NSImage* image =
    [[NSImage alloc] initWithContentsOfFile: imagePath];
```

Drawing the image such that it fills the entire view is simply a matter of calling the following NSImage method:

```
NSSize imageSize = [image size];
NSRect imageRect = NSMakeRect( 0, 0,
    imageSize.width, imageSize.height );
[image drawInRect: viewBounds fromRect: imageRect
    operation: NSCompositeCopy fraction: 1.0];
[image release];
```

In a real screen saver, you'd most likely want to load any images you need in startAnimation and keep them around until stopAnimation or even projectNameView's dealloc method since it is very inefficient to load the image in for every frame. You may even want to provide an accessor method like the following that handles loading an image the first time it is needed.

```
- (NSImage*) backgroundImage {
    if ( backgroundImage == nil ) {
        NSBundle* saverBundle =
            [NSBundle bundleForClass: [self class]];
        NSString* imagePath = [saverBundle
            pathForResource: @"test image" ofType: @"JPG"];
        backgroundImage = [[NSImage alloc]
            initWithContentsOfFile: imagePath];
    }
    return backgroundImage;
}
```

As you move beyond Cocoa, you may require some bit of functionality from Quartz 2D that Cocoa doesn't expose. Not to worry. Cocoa makes it easy to drop down and call Quartz 2D directly. When the system calls your screen saver's drawRect: method, the drawing environment is already set up for you. You can draw immediately with Cocoa as we've seen or you can ask Cocoa for the CGContextRef that you'll need to make Quartz 2D drawing calls. The code to retrieve the context looks like this:

```
CGContextRef context =
    [[NSGraphicsContext currentContext] graphicsPort];
```

Add that line and the following code to the end of drawRect: to draw with Quartz 2D:

```
CGContextSetRGBFillColor( context, 1.0, 0.0, 0.0, 1.0 );
CGContextSetRGBStrokeColor(context, 0.0, 0.0, 1.0, 1.0 );
CGContextBeginPath( context );
CGContextAddArc( context, NSMidX( viewBounds ),
    NSMidY( viewBounds ), NSHeight( viewBounds ) / 4.0,
    0.0, 2 * 3.14159, 0 );
CGContextClosePath( context );
CGContextDrawPath( context, kCGPathFillStroke );
CGContextFlush( context );
```

This code draws a large red circle with a blue outline in the middle of the screen as you can see from Figure 6. If you add this code after the rotating rectangle code from our basic screen saver

and isRotatingRectangles is true, you'll notice that the circles don't draw in the middle of the screen any more. Why not, you ask? Because the Cocoa drawing APIs are layered on top of Quartz 2D, that means that they share the same drawing environment, including the underlying transformation matrix. When the Cocoa code uses NSAffineTransform to rotate the drawing of the rectangle, that rotation also applies to any subsequent Quartz 2D code. The moral of the story is to pay attention to any changes you make to the Cocoa or Quartz 2D drawing state since one affects the other.

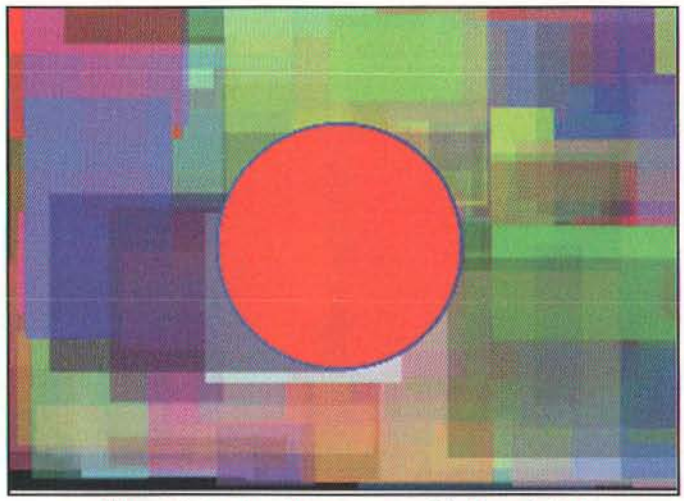


Figure 6: Saving the screen with Quartz 2D

Summary

Well, that brings us to the end of our series on screen savers. In this article we've added a configuration sheet to our screen saver, stored and reloaded the resulting default value, learned how to handle keyboard and mouse events, and investigated NSImage and Quartz 2D. Now you've got the information in hand to go out and write some really cool screen savers.

About The Author

David Hill is a freelance writer living in College Station, Texas. In a former life, he worked in Apple's Developer Technical Support group helping developers print, draw, and write games. In his free time he dabbles in screen savers and other esoteric topics.

Multiple Formats. Multiple Platforms. Complex Installers.

We have the solution:

StuffIt Engine SDK

Solving the compression & multi-platform puzzle!
www.stuffit.com/sdk/

Licenses start as low as $99/Yr.

Put the power of StuffIt to work for you.

- Adds value to your applications by integrating compression and encryption tools.
- The only tool that supports the StuffIt file format.
- Make self extracting archives for Macintosh or Windows
- Available for Macintosh, Windows, Linux or Solaris

Mac

solaris

StuffIt InstallerMaker

An OS X native version ready for developers!
www.stuffit.com/installermaker/

Prices start at $250

Give your software a solid beginning

- Create Macintosh OS X and Macintosh Classic compatible installers
- Get all the tools you need to install, uninstall or update your software in one complete package
- Add muscle to your installers by customizing your electric registration form to include surveys and special offers.

Allume Systems
www.allume.com
email: dev.sales@allume.com

Mac OS X Server 10.4

Tiger Server, An Overview, Pt. 1

Well, Mac OS X 10.4 Server, Tiger Server, is finally here! And there are a ton of changes. Enough so that most administrators are going to want to take their time upgrading, because Apple has given us what is the biggest upgrade since Server 10.0 came out. Literally, there is not one part of Server that has not been changed in some fashion.

Welcome

It's not just existing feature changes and bug fixes either. New stuff, stuff you've never seen before in Mac OS X Server without installing a lot of software and doing a lot of work on your own is here. Chat servers, Weblog servers, Software Update Servers, it's all there. This is one upgrade that's going to be worth the money, on pretty much any level you want.

However, like any kind of review, we need to focus on what we're going to cover, so, for the sake of some vague attempt at not turning this into "Tiger and Peace" we'll let the new server administration tools be our guide.

One thing to note here, by "Server" I'm not just restricting this to Xserves. I'm talking about any machine able to run Mac OS X Server 10.4. If I mention an Xserve-only feature, I'll make sure to note it as such.

Server Admin

Server Admin is the heart of the GUI toolset for managing Mac OS X Server, and as such, is the primary tool of the server administrator. Any changes in Server have to be reflected here, and because Server or not, it's still a Mac, the GUI has to work right the first time. Server Admin is not however a client management tool. That's Workgroup Manager, which we'll be looking at a little later on. Server Admin is only here to let you run your Mac OS X server installations. If you think about Server Admin's reach ending at the network interface, you're on the right track.

Server Settings

One new feature in Server Admin, and one that I'm very happy with, is the ability to turn SSH on and off. With earlier versions, you had to use other applications to remotely manage SSH, such as ARD or physically logging into the server. Considering the advantages of using headless servers, (no monitor), being able to remotely control SSH without having SSH on is a necessity, so making it a checkbox is a welcome feature.

The serial number in Mac OS X Server has changed as well. Each copy of server now checks on the network to ensure that you aren't running multiple copies of server with a single copy serial number. It only checks on the local area network, not back to Apple. While this is inconvenient, it's not going to hurt anyone playing by the rules. As well, if you are going to buy multiple copies of server, and want to automatically set them up

so that you don't have to manually enter separate serial numbers, Apple will provide serial numbers in that case which don't use this feature, so that someone trying to set up a couple hundred servers via remote install doesn't lose their mind.

Date and Time now contains both the Date and Time settings, along with the Time Zone settings, so they're grouped together in a way that makes a little more sense. This consolidation of settings has been implemented throughout Server Admin, so if you're used to Panther's Server Admin, Tiger's will take a little getting used to, but I've found it to be a much nicer tool once I did.

Two changes in this part of Server Admin are how you manage SSL and access to services. In Panther, SSL management was done in the individual service settings. So, for SSL web services, you had to tell the web service about any certificates, set them up, etc. In the email service, your only option was to use SSL or not. If you wanted to do more with SSL in email, you had to leave Server admin and use the command line. In Tiger, SSL is more properly integrated in with the various services that use it. In Tiger, basic certificate setup is done at the server level, and you then tell the services which certificate you want to use. So you can have multiple certificates set up if need be, and then just select the certificate you wish to use within the service. If you want or need to use a service - specific setup that is different from the certificates you've set up in the main SSL settings, you can do that too.

ACLs

Mac OS X 10.4 Server has changed how you set access to various services, and this revolves around one of the biggest changes in Tiger as an OS, namely Access Control Lists, or ACLs. ***See Ed Marczak's May Mac In The Shell for more great coverage of ACLs!***

With Panther and earlier, permissions were simple. Every file, or folder had one owner, one assigned group, and everyone else. Each file or folder had three options, read, write, or execute. (In Unix, when you list the contents of a folder, you are 'executing' it.) While this is simple, and reasonably universal, it's also very limiting. To change access for various people, you have to start getting complicated with group memberships. Of course, if you get too complicated, then you run into problems, because you can only belong to 16 groups prior to Tiger. (If this sounds like a lot, it really isn't. The first user on an OS X system belongs to two groups by default, their own, and admin. In a university or business environment, you can hit the sixteen-group limit with ease.)

Aside from the group limit, three permissions caused other problems. If you have write access to a file, but not the folder, while you couldn't delete the file, you could open it up and delete every bit of data in the file. Even with other options, like the sticky bit or the chflags command, the traditional Unix permissions scheme became very hard to administer if you needed to use things in a way that they simply weren't designed to handle.

Enter ACLs. ACLs approach permissions differently, and allow for greater flexibility. An ACL is just that; a list that defines who can access a given file or folder, and how. The big advantage to ACLs is that who can access a folder, and what they can do is not as limited. For example, let's say I have a folder and I have three groups of people who need different access to a folder, and the contents of the folder. Under Panther, I'd have to do a lot of workarounds to deal with multiple groups needing different access. With Tiger and ACLs, I can just assign each of the three groups to the folder and files and set their access accordingly. I can set individual user access separately too. I can also specify how inheritance works, too, so that a given ACL can apply to every subfolder a given user or group creates, and all the items that will ever be inside those subfolders, only the immediate children of that folder, or I can kill inheritance entirely.

Write access is now no longer a binary issue with ACLs. I can specify that a user can delete a file, but they can't modify the contents. I can allow a user to administer the permissions on a folder without making them the owner, and without making them an admin. I can have someone who can modify existing files but can't delete them. While smaller entities may not need this level of control, even a company with only a hundred users or so can easily get into some pretty complex rights management issues, and Tiger allows you to manage this outside of any sort of third party software.

Now there are some caveats here. Officially, ACLs are only supported on Mac OS X Server, and you have to use Workgroup Manager to manage them. You can observe ACLs in the Finder, but you can't set or change them. For most, this will work. However, if you want command line access to ACLs, you can use `chmod` to set ACLs, and `fsaclctl` to enable or disable ACLs on a given volume. By default, they are enabled for Tiger Server, and disabled in client. While you can enable ACLs in client, if you don't need to, I'd recommend not doing it. You can get very complex permissions structures with ACLs, and you can accidentally open some large holes. It's also going to take a little bit before third party software knows what to do when a user has write access but not delete access. (This comes into play if a file saves by creating an entirely new file, and deleting the old one. With the finer - grained controls of ACLs, it's trivial to break this if you aren't careful.)

A part of ACLs, are the Access Control Entities, or ACEs. An ACE is part of an ACL, and is the actual line in the ACL that specifies what a user or group can do. (The best way to think of it is that an ACL is a list of ACEs, and

each ACE lists a single person or group's Access Control setting(s)). The ACE contains 4 items:

1. User/Group: Who the ACE refers to, identified by a 128-bit Universally Unique ID, (UUID) number. UUID's avoid the problem of "Which John are we talking about"
2. Permission Type: Is this ACE allowing or denying a set of permissions
3. Permission: Which of the 13 possible permissions are being used and how
4. Inherited: Is this ACL inherited from its parent, or explicitly set

Explicit ACLs are set manually in Workgroup Manager or via chmod. With the Tiger Server ACL model, there are four kinds of inheritance. "Apply to this folder" means just that. They only apply to the specific folder. "Apply to child folders" means apply these settings to subfolders, but not files. "Apple to child files" means apply these settings to files in the folder, but not subfolders. "Apply to all descendants" means apply these settings to everything. (Note that if you really want "Apply to all descendants to apply to ALL descendants without question, you have to check all of the last three options.) While it may seem silly to have to specifically apply permissions to the current folder, there's some sense. In the case of a school, you can have a master school folder that only administrators have access to, but a series of class(room) folders that have different permissions and inheritance settings. (I said it before, and I'll say it again, be ***careful*** with ACLs. There are 98,304 possible combinations of ACL permissions, ***before*** you take complex folder hierarchies into account. You can create some really bizarre problems with them if you just slap them on like paint.)

When you use ACLs, you also have to keep in mind that there are rules of precedence for ACLs/ACEs:

1. If an object has no ACEs, then traditional Unix, (or more correctly, POSIX) permissions are used
2. If an object has multiple ACEs, then Tiger Server starts at the first one, and works its way down the list until the requested permission is allowed or denied. It then uses that ACE, along with the POSIX permissions, to determine access
3. Deny permissions overrule Allow permissions. So if someone has allow as part of a group and deny as an individual, the Deny wins. Tiger Server also reorders ACLs so that Deny rules come first. (Did I mention that ACLs can get complicated?)
4. Allow permissions are cumulative, so Allow permissions are the union of all allow permissions, including POSIX permissions.

In addition to local file/folder ACLs, both Apple File Protocol, (AFP) and Server Message Block, (SMB) file sharing protocols support ACLs, and you have to be using HFS+ to use ACLs.

Finally, Tiger's ACLs are compatible with Windows 200X ACLs. While this may not make many in the open source community happy, for anyone working in an Active Directory environment, this is huge, since you can now use your Active Directory tools to manage permissions on Tiger Server Windows share points. As we will see in the Windows section, this creates some really cool capabilities.

However, those are just file system ACLs. With Tiger Server, we also get Service ACLs, or SACLs, set in the Access tab of the main server settings in Server Admin. SACLs allow you to split out access to various services in Tiger Server. So you can, for example, only allow your domain admin groups to physically log into a server and use SSH to that server, but allow anyone to use file sharing via Apple File Protocol, or AFP. While this could be done in Panther, it was a manual process, and tended to complicate things. Tiger greatly simplifies this. Just like with file ACLs, you can add multiple groups or users, however, with SACLs they'll all have the same setting, so there's less of a point in doing this in general.

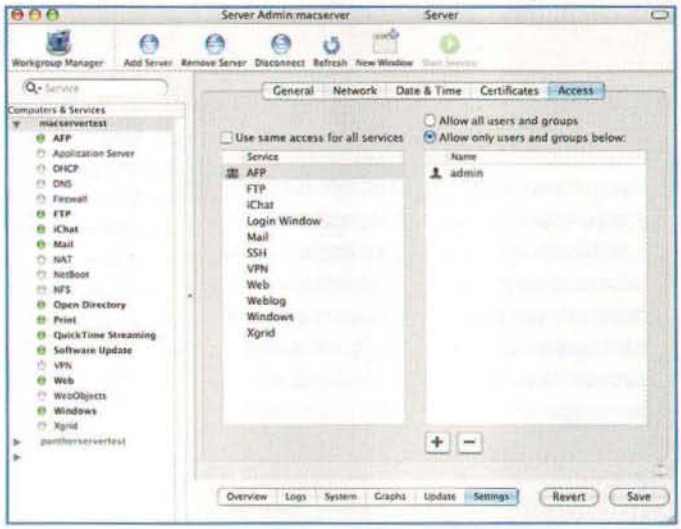


Figure 1: Service ACLs in Server Admin

AFP Service

With Tiger Server, AFP is now at version 3.2, and includes a few new features such as support for Unicode file names, ACLs, and 64-bit file sizes. The maximum AFP volume size is 2TB, which, while well under a 64-bit size, is still a good size for a volume type that you'll only see over a network. The truth is, AFP is a mature protocol that already supported Kerberos, SSH tunneling, and most of the other features that Mac users need, so there's not much ***to*** change in Tiger, and Apple has wisely chosen to leave that which works alone. The one complaint I do have is that you can't force SSH connections, which is a nice option for those with a need for high security.

DHCP Service

DHCP as a protocol hasn't changed much in quite a while, so there's not a lot for Apple to change at the protocol level. From a management point of view, the big "new" feature is the ability to assign static addresses to a specific machine. (While

you could do this in Panther, Tiger integrates this into Server Admin.) While static addresses may seem to be at odds with the idea of DHCP, it really isn't. Static addressing allows you to automate router assignment, subnet mask settings, DNS and LDAP settings, but still keep the IP address the same. This can be useful for servers, test machines that you want to always have the same address without having to manually assign the rest, etc. It's also handy for shops running their own DNS without the benefits of Dynamic DNS, aka DDNS. The client is still set to DHCP, they just simply always get the same IP address is all. One thing to note here, if you have a machine that uses multiple interfaces, like Ethernet and AirPort, you'll need an entry for each interface on that machine, since static DHCP uses the Media Access Control (MAC) address to identify the machine, and every network interface has a unique MAC address.

DNS Service

DNS is, in many ways, the most critical service in a Mac OS X network. While the Apple documentation talks about it in polite terms, I'll say this up front: ***If your DNS is not working correctly, you are going to have nothing but pain with Mac OS X Server. Period.*** A badly configured DNS will not only cause you problems, but can, and quite often will, reach out and cause problems for people all over the Internet. DNS is a place that you do not ever want to go without a good solid guide, and while Apple's documentation is good for enabling and managing DNS on Apple servers, you really, ***really***, REALLY want to buy and become one with the O'Reilly book on DNS and BIND. It is the "bible" for DNS administration, and no one running their own DNS should be without it.

There, that's out of the way. In Tiger, Apple is using version 9.2.2 of the Berkeley Internet Name Domain, (aka BIND. DNS is the service humans use, BIND is the server that implements those services. BIND is also not a single process, but rather a collection of services, including the name daemon, or named.) While the version of DNS/BIND that Apple uses should normally support Dynamic DNS, (DDNS, that is, DNS names tied to your machine, not your IP address, so even if your address changes due to DHCP, you still have the same DNS name), there's no support for this in Apple's implementation, so if you want to use DDNS on a Mac OS X Server box, you have to install and configure DNS on your own without Server Admin. This is, by the way, a glaring deficiency in Apple's DNS implementation, and something that will make many network administrators choose a different OS base for their DNS services.

The only real change in Server Admin here is the addition of a Secondary Zones tab, which makes setting those up and monitoring them a little easier.

IP Firewall Service

Tiger's firewall has gotten quite a few updates over Panther's, although it is still based on IPFW. One change is semantic in nature, but just enough to make talking to someone remotely a bit confusing is that where Panther called firewall entries "Filters", Tiger calls them "Rules". Same thing, different name, but just enough to drive you insane on a remote support call.

The UI in Server Admin has changed quite a bit though. One change is that Active Rules are no longer shown on the first screen of the Firewall section. Instead, you get basic statistics for the firewall operation. Active Rules now have their own tab.

Setting up the Tiger Firewall is different as well. There is now a "Services" tab, which contains presets for the services that are running or can run on the Server. Address groups now have their own section as well, as opposed to Panther, which combined these two functions in the "General" tab. The Services tab in Tiger has far more entries, including explicit entries for Apple Remote Desktop's (ARD) functions, both 2.x and 1.X, as opposed to Panther, which only has a default entry for ARD 1.2's single port setting. This allows administrators quick access to enabling or disabling various OS X services with ease. One welcome addition is that iTunes Internet Radio streams have their own entry, (Ports 42000-42999 TCP), so allowing local network iTunes sharing but disabling Internet Radio streaming is a snap. A new feature of the Services tab is that you can create, edit, and delete Service entries in the GUI, something that you could not do in Panther. (You can't do it with Tiger's Server Admin tools against a Panther server either.) This allows you to add your own service entries without having to do so in the "Advanced" tab.

The Advanced tab hasn't changed its basic functionality much. It's still the place where you create very specific rules for the firewall. The UI does now show you which rules are locked and not editable in the GUI, a very welcome change from Panther's "surprise" mode for such things. The other changes here are controls for "Stealth Mode". When enabled for TCP, UDP, or both, a computer attempting a connection on a closed port doesn't get a failure message. Instead the packets for that port are dropped, and for all practical intents, you don't exist. This is a nice addition to the security features of Tiger, since if a server isn't seen it's much harder to attack. Even a negative answer is a positive answer if all you care about is the answer's existence, and not the content.

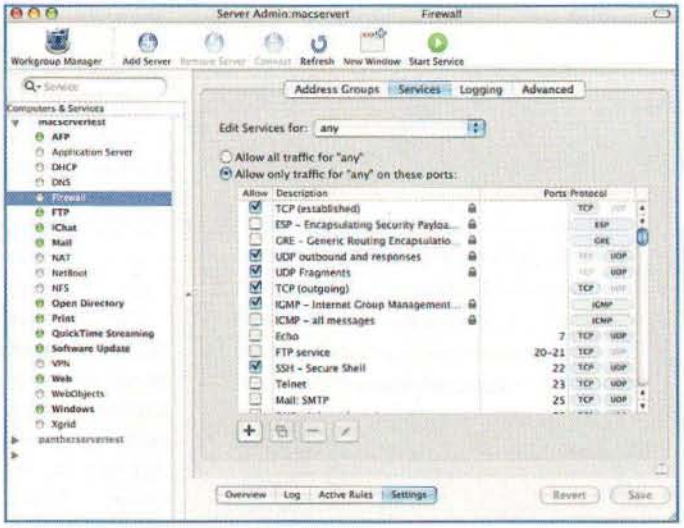


Figure 2: The new Firewall Services tab

One welcome addition is not in Server Admin, but in the documentation for Network Services under Tiger, where Apple gives you step-by-step instructions on how to reset your firewall back to the defaults, in case you did something silly, like kill all external network access to your server. For a new admin, this is something they will need, and having it documented is A Good Thing. Another change that may or may not be welcome is the elimination of the sample command line rules from the IP Firewall Service documentation in the Network Services manual. They're still documented, but are now part of the newer command line reference manual.

FTP Service

Tiger's FTP service is, like FTP, essentially unchanged. The only "new" feature is that the FTP service enforces ACL settings. Other than that, it's what it was in Panther.

iChat

The iChat server is however, completely new to Tiger, and has no analogue at all in Panther, so we don't have to care about any changes. The iChat server in Tiger is not an AIM server, but rather Jabber server. Jabber is, according to the Jabber organization, (at http://www.jabber.org/)...***best known as "the Linux of instant messaging" — an open, secure, ad-free alternative to consumer IM services like AIM, ICQ, MSN, and Yahoo.*** Jabber is also extensible, so, with a little work, you can use Jabber to talk to pretty much every IM network on the planet.

Setting up the iChat server in Mac OS X is, for basic functionality, ridiculously simple. You set up the Host domain, usually the DNS name of your server, set up a welcome message, and decide if you want to use SSL. If you do, you select the certificate you want to use, save your settings and click the "Start Service" button. That's it, you're now an iChat server. Jabber supports all the basic IM functions, but not every client supports every function the same way.

Out of the box, iChat in Tiger can (obviously) hook up to an iChat Server. As long as you have an account on the iChat Server, you can log into the iChat Server. You just create a new Jabber account to do so. When using Jabber, there are some unique aspects of Jabber to be aware of. First, your Jabber userID looks like an email address, and is ***shortusername@ichatserverjabberhostname*** So, if your short user name is jwelch and your iChat server name is jabber.mactech.com, then the Jabber user ID is ***jwelch@jabber.maccentral.com***. Since Jabber doesn't support Kerberos as a standard feature, and there are, as a result, not a lot of Jabber clients that support Kerberos, you can't use the iChat Server in a single-signon environment. What this means is that when you're setting up Jabber clients, whether iChat or not, you're going to have to enter the password into that setup, even though it's the same password you use in Tiger Server's single signon environment.

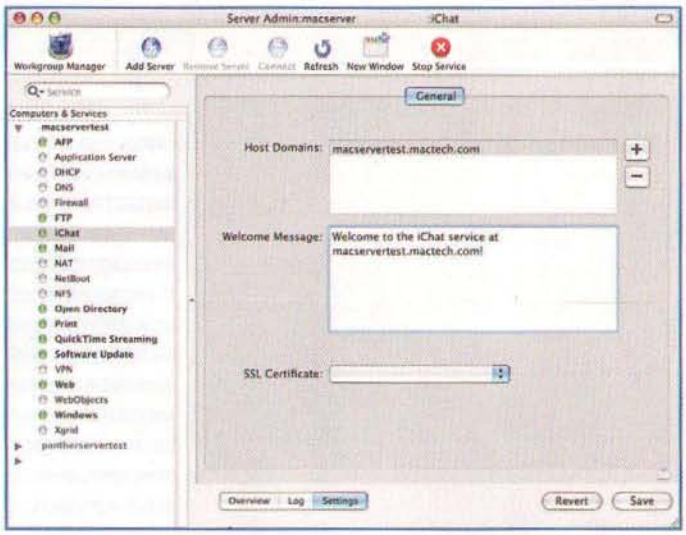


Figure 3: iChat Server Settings

Another caveat is that if you use iChat, the only version of iChat that supports the iChat server is version 3.0, so if you're using an older version of iChat, you'll have to use a separate Jabber client to talk to the iChat Server. While the iChat Server supports using SSL for security, it's not all encompassing. Only the text conversations are encrypted. Your login session, file transfers, and any audio or video chats are not encrypted. So if you need an extremely high security setting for IM, you're going to have to do that work yourself.

As I said before, not all Jabber clients support ever Jabber configuration the same. So sometimes, even a really neat Jabber client won't work with every feature of iChat's Jabber implementation. For Windows and Linux, the client I found that worked the best with the iChat Server was Psi, (http://psi.affinix.com/). It was easy to set up, and supported file transfers well. PSI's interface is a bit too ICQ-ish for my tastes, but it works well. Psi runs on Mac OS X/Windows/Linux, so if you want a "standard" environment, it's an option. It supports GPG and PGP for file encryption, one way of dealing with the iChat Server's lack of security here. If you want a pretty client for Windows/Linux, Gush, from 2entwine is an option. It doesn't support file transfer, but it's very pretty, which is always nice.

For Palm device users, your best bet is Chatopus, (http://www.chatopus.com/), which supports SSL connections. (I wasn't able to test Chatopus since I don't have a Palm device. However, anyone wishing to send me a Treo 650, feel free to do so, and I'll happily test it for you.)

For Pocket PC/Windows Mobile devices, I found imov, (http://www.movsoftware.com/products/imov/imov.htm) to be a good choice. It works well with iChat server, and in my tests on a Sprint PPC-6600, was easy to set up and use, although file transfer isn't supported. This is probably not a major problem for most PDA/Smartphone users. There are a number of clients for the Blackberry and Symbian devices, but I have no idea about which ones work best. (Although like my Treo offer, if you want to send me a BlackBerry or a Sony p910/Nokia 93XX device, I'll test it for you, no problem)

Data Banks Communications Inc

HOME OF THE FREE COMPUTER
THAT'S RIGHT FREE COMPUTER!

GET A MAC MINI OR DELL COMPUTER FREE JUST SIGN UP FOR VOIP DSL BUNDLE

INFOSPEED 768K/128K	$ 69.95
INFOSPEED 1.5M/128K	$ 69.95
INFOSPEED 1.5M/384K	$ 69.95
INFOSPEED 7.1M/768K	$ 199.95
INFOSPEED 384K/384K	$ 69.95
INFOSPEED 768K/768K	$ 149.95

TeleSoho 1.5M/128	$69.95
TeleSurfer Plus 608K/128K	$69.95
Residential 768K/128K	$69.95
SOHO1.5M/768K	$79.95
SOHO3.0M/384K	$79.95
SOHO3.0M/768K	$79.95
SOHO6.0M/768K	$139.95
TeleSpeed 144K/144K	$125.95
TeleSpeed 192K/192K	$125.95
TeleSpeed 384K/384K	$149.95
TeleSpeed 768K/768K	$159.95
TeleSpeed 1.1M/1.1M	$199.95
TeleSpeed 1.5M/1.5M	$249.95

Ask for business line special!

www.databanksglobal.com

For questions regarding sales or service, please call us 24 hours a day at 1-866-624-6114

One thing the iChat Server doesn't do is automatically log the contents of IM conversations. This can be an issue for companies in this day and age of Sarbanes-Oxley and other regulations. The only way I could find to do this is a third party plugin, Bandersnatch, from Funky Penguin, (http://www.funkypenguin.co.za/tiki-view_articles.php). It's not a dead simple setup, but if you have the need, and some basic MySQL skills, then it can log all iChat Server conversations. (This is one of those things that can be used for good or evil, but there are cases where it's necessary, and in that case, you either use it, or don't use iChat Server.)

Mail Service

While not new in Tiger, the Mail Service has been the recipient of a number of changes, fixes and new features at almost every level. Two critical new features are the integration of SpamAssassin and ClamAV for anti-spam and virus detection. While there are still no native Mac OS X viruses, and less than a handful of trojans, when you run an email server, you don't know what platform is going to be connecting, so it's better to have extra protection than not enough. Virtual Hosts are now supported in Server Admin, so you aren't forced to the command line for this. This is something of no small importance to ISPs. Sieve scripting, for creating server-side mail rules is supported in Tiger, and even documented. While you could implement Sieve in Panther, you had to do it from third - party documentation. Server rules are an important part of enterprise mail services, so it was important that Apple do more for things like Sieve.

For administrators with large email setups, you can now split mail stores across multiple partitions or even remote filesystems mounted locally on the server, so that a single hard drive incident doesn't cause a loss of service for your users, or so you can load - balance the storage duties of user email. Tiger's mail service also supports the new Service ACLs, so you can have better control over who has access to the email service on any given server. This can create some confusion as to what a given user can really do when you have separate user account settings in Workgroup Manager for mail access. The simple rule of thumb is, if a user has email service enabled either in Workgroup Manager or in the Mail SACL in Server Admin, then they have access to Tiger Server's Email service. If you want to make sure a user doesn't have access, then they have to be denied access in both locations.

With Tiger, you now have two options for email aliases. You can either use Workgroup Manager's ability to create login aliases and use those for email aliases too, or create them manually in /etc/postfix/aliases. The biggest difference between the two is that aliases created in Workgroup Manager won't work with Sieve scripts. Yet Another New Tiger Email Feature is that you can now manage mail quota handling in Server Admin.

Postfix, Cyrus, and SquirrelMail are still used for SMTP, POP/IMAP, and Webmail respectively, so there's no changes to the root servers that make up the email service beyond version numbers.

The Mail service module in Server Admin has been redesigned up and down to give Mail administrators more power outside of the Command line. Along with this greater power, the Mail service documentation has been extensively revamped and rewritten so that mail administrators can get a lot more use from Reading The Fine Manual. The mailing list section alone goes into more detail in its first section than Panther's documentation did for the entire mailing list entry, including solid information on what you use the web interface for as opposed to Server Admin, a welcome change and an important improvement to Tiger Server.

As I've been noting, Server Admin's Mail Service module has been extensively revamped, to give the GUI the kind of power that an email admin needs. Outside of new settings, there's a "Maintenance" panel, that allows you to handle basic mail, well, maintenance functions. So things like the mail database repair function can be run from Server Admin with a click. The Database tab here also allows you to view basic information on all the mail stores a server is using, (important, since Tiger lets you have multiple mail stores in multiple locations). Mail Queue management is another feature of Tiger, allowing you to view the current mail queues and either delete or retry messages that are having problems. The Migration Tab allows you to import any Mac OS X Server 10.1 or 10.2 mail databases from Server admin, giving you the option of migrating selected users or every user.

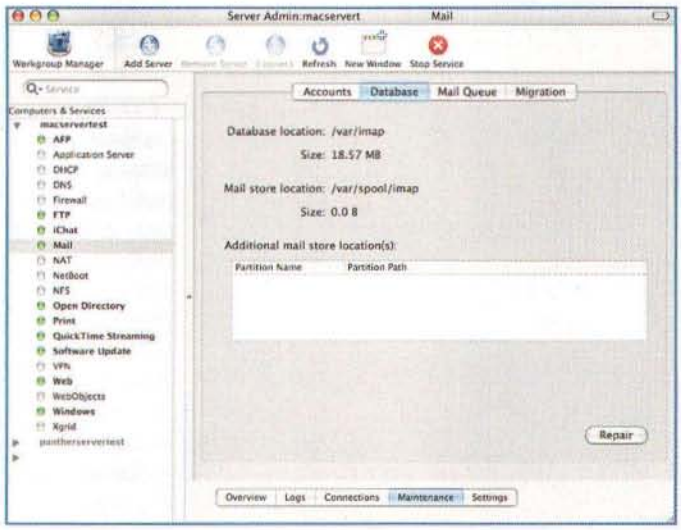


Figure 4: Mail Database Maintenance tab

The Settings panel has a metric ton of new capabilities, all aimed at making the GUI more useful. In the General tab, you can now set the mail server to deliver to /var/mail when you turn off POP and IMAP services. This can be useful if you need to receive mail without delivering it, and not cause email to bounce. The controls for enabling SMTP are also where you specify the domain and host names for the email server. Along with enabling SMTP/POP/IMAP, you can now separately start or stop incoming and outgoing mail here. This is a real convenience for everything from troubleshooting to dealing with network congestion problems.

The Relay tab is new, and has the functions of the "Filters" tab in Panther's Server Admin. With the separate anti-spam and anti-virus capabilities of Tiger, and the fact that the settings in this tab are more focused on relaying settings, adding this tab just makes for a more logical and more intuitive UI for administrators.

The Filters tab is now where you control your anti-spam and anti-virus settings. SpamAssassin's junk mail score setting, accepted languages, locations, what to do with junkmail, how to handle it if you forward it are all here. The Virus settings are pretty simple: Turn AV on/off, what to do with infected email, and who to notify of infected email. You can also set the frequency of daily updates to your anti-spam and anti-virus databases here. Once a day is probably the slowest you'd ever want to set this.

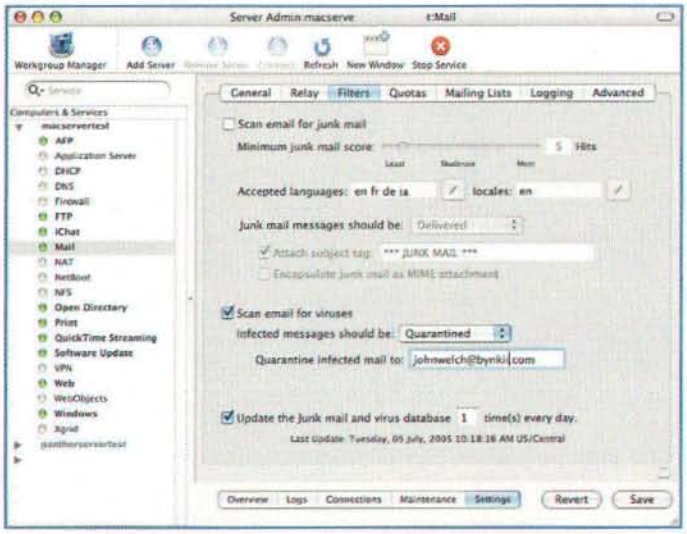


Figure 5: Mail Filters tab

The Quotas tab is a new feature in Tiger, and works with the Quota settings in Workgroup Manager. This is where you set things like maximum incoming message size, and how to handle users near quota or over quota.

The Mailing List tab is unchanged, an island of familiarity in the newness of Tiger's admin tools. The Logging tab adds detail level settings for the Junk Mail and AV features, along with finer control of the logging levels for SMTP logs and POP/IMAP logs.

The Advanced tab is now split into three sections. The security section is where you set your authentication methods for SMTP/POP/IMAP, and your SSL usage for SMTP and POP/IMAP. SSL usage now allows you to specify the certificate used, instead of just the Use/Don't Use/Require settings in Panther. The Hosting section is where the Virtual Domains and Virtual Host settings are contained. Finally, the Database section is where you specify the mail database location, the main mail store location, and the location of any additional mail stores.

Keep in mind that any command line options you used in Panther are still there. None of the GUI improvements are coming at the expense of the UI. You just now have a better GUI to use along with the command line if you want.

NAT Service

Like FTP and AFP, Network Address Translation hasn't changed much in concept or practice in a while. Server Admin in Tiger adds the convenience of being able to set the NAT service to just act as an IP Forwarder, or to perform IP Forwarding and NAT.

NetBoot Service

While Tiger doesn't do anything radical to NetBoot/Netinstall, there are some small, yet welcome changes. The number of AFP connections has been increased to whatever the resources on your server limit it to, nice for those administrators with high-powered setups. Network install images can now be created as block - copy images instead of file - copy images, which result in much faster installation speeds. The previous command-line only option for setting up images on remote servers is now a part of the GUI tools. Images can now be created with Directory Services settings configured in the image, so that as soon as the machine boots, it's configured correctly without needing to use the DHCP LDAP settings. (You could do this in Panther, but you had to copy /Library/Preferences/DirectoryServices/ from the machine you were creating the image from to the image. In Tiger, this is integrated into the toolset.)

Since so much of NetBoot/NetInstall revolves around image creation, we should start with a look at the new features in the System Image Utility, formerly the Network Image Utility. One thing administrators will notice right off is what's missing: Information on creating Mac OS 9 boot images. Considering how dead OS 9 is to Apple, along with the fact that Intel – based Macs can't run Classic, much less boot OS 9, this is no surprise, but it means that if you still need to create new OS 9 install images, you should consider keeping a machine running Panther Server around.

The basic functions of the System Image Utility haven't changed, but the implementation has been expanded somewhat. You can now explicitly assign the protocol to use to serve the image, although NFS is the preferred method still. You can, as I said earlier set up a machine as an image server and let NetBoot serve from there. You have to create the image locally, and manually copy it to the specified location, but once it's there, you can serve it from there. While this would seem to make it an ideal candidate for an Xsan and a few front end Xserves, that's not the type of use an Xsan's optimized for. That's not to say you physically cannot do it, but you would ***really*** want to test this out first before buying the gear. A new option for install images is the "Change ByHost preferences to match client after install" feature, which allows administrators to create ByHost preferences, (preferences tied to the Media Access Control, or MAC number of the main network interface card.) This takes the ByHost of the image source, and when pushed down to the client, will change the preference to match the client. Considering that preferences like screensaver settings, iTunes

settings, .Mac and Classic settings, et al use ByHost, this saves a lot of work on the back end of the install for administrators.

One new trick that rates high on my "cool list" is the Model Filter setting for both install and boot images. This allows you to tie an image to specific models of Mac. This is very important in situations where you have different images for portables versus desktops. This avoids someone accidentally or deliberately installing the wrong image on a type of hardware that you don't want it installed on.

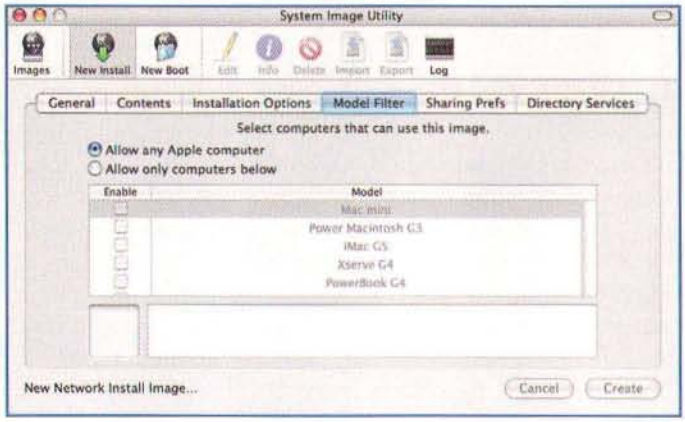


Figure 6: System Image Utility Model Filter tab

Another problem that Tiger makes easier to solve is naming imaged machines. If you have to image a dozen or so machines, giving them all unique sharing names can be tedious. Tiger's System Image Utility gives you two options right in the UI for doing this. If you want them all to have the same name, with the MAC address grafted on then you just enter that name (sans MAC address) in the "Computer Name" field in the "Sharing Prefs" tab for boot or install images. If you leave it blank, then they either get -AUTOMATIC- or DNS names from the network based on their IP address. You can also create a tab-delimited text file that has the desired name for each machine and the corresponding MAC address, and tell the System Image utility where the file is in the "File Path" field. When you image the machines, if they have a MAC address that's on the list, they get the name you give them. If they don't have an address in the list, then they get the name from the setting in "Computer Name".

Once you've created the image, then you can use some of the new tricks in the NetBoot service to make managing them easier. One thing administrators love is information on our servers. We love knowing all the details, and we hate having to spend a lot of time sussing them out. Server Admin in Tiger does that for NetBoot far better than Panther did. The first screen in the NetBoot service tells you which images are enabled, and if the necessary protocols are running for each type of image, (AFP/NFS/HTTP/DHCP).

Monitoring NetBoot clients is easier with Tiger too, as you get, in addition to all the info that Panther provided, information like the System Type of the client, handy if you have a couple thousand machines and haven't yet memorized every detail of every system.

The settings haven't changed much. You can set the maximum number of AFP connections, you can have Server Admin look up MAC addresses for you based on the client's IP address or DNS name, and you can set the logging level.

NFS Service

Tiger's NFS service is generally unchanged from Panther's. If you want to do more than turn it on/off, select the number of daemons, and the connection type, you're going to need to use the command line and become one with the various NFS books.

Open Directory Service

There are many, many changes in Open Directory, especially in the areas of Windows integration, and they are both welcome, and needed. Open Directory is literally the heart of Mac OS X Server. Every service, every feature at some point deals with Open Directory, so improvements here ripple out into every other part of Tiger Server. However, it's important to remember that Open Directory, (OD) isn't just one thing. It's not like Microsoft's Active Directory, which is a single, albeit, complex product. Open Directory is more of a collection of technologies, integrated so that they all work together for Mac administrators. By default, Open Directory is primarily OpenLDAP on a BerkeleyDB datastore, with a MIT Kerberos Domain Controller for primary authentication and Single-Signon.

But OD is not just that. In fact, it's really nothing more than a container. LDAPv3, NetInfo, Active Directory, NIS, etc. are all just plugins to OD. Apple provides a base set of plugins with Tiger, including: Active Directory, AppleTalk, Bonjour (nee Rendezvous), BSD Flat Files & NIS, LDAPv3, NetInfo, SLP, and SMB/CIFS. You can also get a number of third party plugins that add features, like direct support for Novell's eDirectory, or provide additional features for things like Active Directory that Apple's plugins don't give you.

With Tiger Server, Apple has further refined Open Directory so that it provides not just improved features, but better security as well. Improvements include things like Trusted LDAPv3 binding, where not only does your system have to authenticate to the server, the server has to prove it's the right server to your system. This is a critical feature, one that makes using directory services a much more secure proposition. With this, and the various features in Mac OS X 10.4 Server, such as Kerberos signing and SSL encryption of Open Directory server communications to the client, doing things like creating a bogus Open Directory Master is effectively impossible, since the server will be unable to authenticate itself to the client. (If the only authentication is client to server, then creating a bogus server that can root your clients is much simpler. It's not child's play, but it gets a lot easier.) Now, if you implement Trusted binding, then you can't use the DHCP discovery of Open Directory Masters, but until there's a standard for trusted DHCP, that option isn't a great idea anyway. The only reason to use the DHCP Option 95 LDAP server discovery was so that you didn't

Extreme Macs

From legendary Apple guru, Scott Knaster, find out all the coolest hacks, tweaks, and mods that you can make to your Mac®, iPod® or even to Mac OS® X Tiger™

EXTREMETECH™

Available wherever books are sold.

WILEY
Now you know.

have to set up directory bindings on each client. However, since the new NetBoot/Netinstall features in Mac OS X 10.4 take care of this, that's no longer as much of an issue.

Active Directory integration is much better, thanks not only to ACLs, but also to support for using AD info for home directories, and better directory information mapping between OD and Active Directory. Along with that, many of the options for Active Directory that were formerly only accessible via the dsconfigad command line tool are now in the UI. Binding a Tiger Server to an Active Directory realm is far simpler, and finally gives you the benefits that it should have in Panther. You can store the LDAP scheme in the directory for convenience, and improved replication.

Looking at the new Directory Access application, in /Applications/Utilities/ there's some obvious changes, mainly the renaming of "Rendezvous" to "Bonjour", which keeps Tibco off of Apple's back, and still keeps the company's Francophilism firmly intact. (Personally, I thought *Achtung!* would have been better, but only if Apple also bought the rights to "Hogan's Heros")

The Active Directory plugin, now at version 1.5, as I said earlier, has had a GUI revamp. Options that were previously only settable or readable from the command line are now in the GUI, such as the mount style for network home locations and whether to force local home directories. New options allow you to set the default shell for a user, more fine-grained control over attribute mapping, using the UNC path in Active Directory settings for a user to set the network home directory location, and a much nicer UI for entering in Active Directory administrative groups. AppleTalk is still the same, and like before, only controls your ability to browse via Appletalk protocols. (AFP file transfer in Mac OS X 10.4 should be assumed to be AFP over TCP/IP only.)

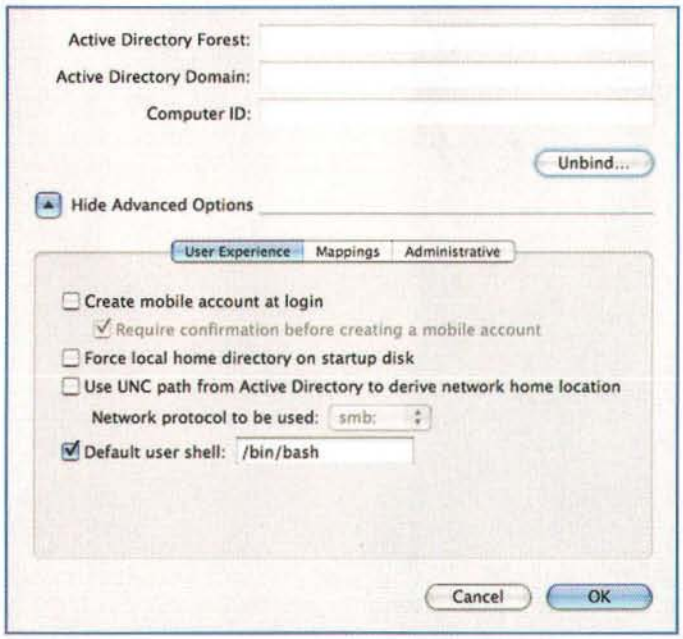


Figure 7: Directory Access Active Directory Settings

The BSD Flat File and NIS plugin has received only minor adjustments, but as that is clearly a legacy technology to Apple, this isn't surprising.

Bonjour/nee Rendezvous Service Discovery is now permanently on in Directory Access. If you want to disable that, you can't do it from the GUI. Note that this is not enabling/disabling all the Zeroconf technologies that Bonjour encompasses, just the ability to find other Bonjour services.

Given all the changes in Mac OS X 10.4 Server's Open Directory infrastructure, one would expect that the GUI for LDAPv3 in Mac OS X 10.4 to have radically changed. But it hasn't, instead getting some minimal changes to improve usability and account for new features. Setting up an LDAP server is more automated, and the UI for options is clearer. For example the initial screen when you add a new server lets you set up SSL, authentication and contact options right away. The manual options have new features as well.

Figure 8: Adding an LDAP server to Directory Access

The Connection tab now lets you set rebind attempt periods and idle timeouts, and gives you the option to ignore server referrals, (Server referrals are where the LDAP server you connect to can tell you to look on other LDAP servers for different information. If you have a lot of referrals or a wonky network, this can greatly increase your login time.) or use LDAPv2 in read only mode.

The Search & Mappings tab adds the ability to save custom mappings out as a template so you can get to them easier next time you need to.

The Security tab is new to Mac OS X 10.4, and incorporates the "Use authentication when connecting" option in Panther's Connection tab, but adds in the new options for the trusted binding options. Within the new Security Policy options, you can avoid rogue LDAP servers compromising your network. The new options are:

- Disable clear text passwords: This one's obvious. It means that you can't use unencrypted passwords to communicate with the LDAP server. From a security POV, this one should be a default.
- Digitally sign all packets (requires Kerberos): If you are using a wide area LDAP implementation, this option is critical. It allows you to ensure that the packet you get is the packet that was sent, unmodified and unmolested. If the packet is changed enroute, then the digital signature will be

wrong, and the packet won't be accepted. This option requires that you use Kerberos.

- Encrypt all packets (requires SSL or Kerberos): Again, fairly obvious, and goes along with disabling clear text passwords. This ensures that all data between client and directory server is encrypted.
- Block man-in-the-middle attacks (requires Kerberos): This option works with the digital signing option to prevent someone from setting up a rogue server posing as an LDAP domain controller.

Note that you can override any of these options at the server level, which is always a good idea on a managed network.

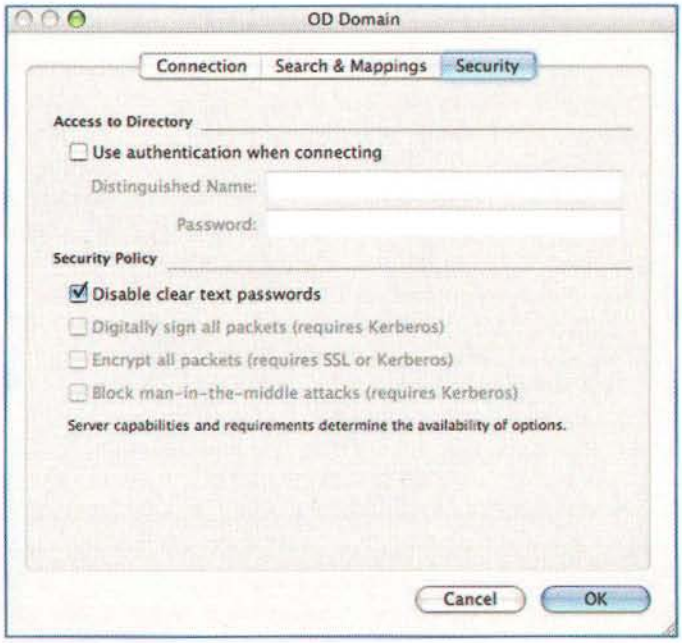


Figure 9: Directory Access LDAP options

I know that some of you are thinking "Well, I've got a really good firewall, why should I worry?" Firewalls aren't magic. For example, do you require every machine on your network to meet a specific security profile before it can use your network? If so, how do you force that? You can do this with Windows, but Mac OS X support for this kind of thing is still as yet unimplemented. A single badly configured machine inside the firewall can leave you open to attack. Firewalls also don't do much about disgruntled/paid off employees. This is not to say that firewalls are not good, useful things, but that they are only a part of a good security policy. Avoiding unencrypted, unsigned data transmissions wherever possible only enhances your security, and gives you additional layers of protection.

Server Admin has new features to make a Mac administrator smile as well. The Log tab in Server Admin's Open Directory settings in Mac OS X 10.4 Server adds in the kadmin & kdc kerberos logs, a Password Service Replication log, and slapconfig log. This helps answer one of the most frustrating things about Mac OS X 10.3 Server. Logs are the best way an administrator has to troubleshoot problems on their network. The fact that Mac OS X 10.3 didn't log Kerberos - specific events by default was one of the most frustrating things about that version. Kerberos troubleshooting can be extremely tricky on a good day, and no logs are ***not*** the way to a good day by any definition. There's also a search/spotlight window built into the Log tab, (indeed, it's in all the Log tabs in Server Admin), which allows you to filter a log for specific conditions. Another new, and very welcome application - wide change in Server Admin is the inclusion of the path to the log you're viewing, so if you want to look at a log outside of Server Admin, finding the specific log file is much simpler.

The Archive tab is a new Mac OS X 10.4 feature for Server Admin. It allows you to quickly back up and restore Open Directory Master settings, including:

- LDAP directory database and configuration files
- Open Directory Password Server database
- Kerberos database and configuration files
- Local NetInfo domain and shadow password database

This is designed as a supplement to using a proper Open Directory replica setup, or in cases where replicas are not possible or practical.

The Settings tab has gained new capabilities, particularly with regard to security and joining other directory systems. In my tests, joining a Windows 2000 Active Directory realm was a three or four - click process in Server Admin, and about 3 minutes in Directory Access and you were done. Once you've finished that, it just works, no fuss, no muss, with far less pain than Mac OS X 10.3, and thanks to new Windows features and ACLs, you get a much nicer level of integration than ever before. The "Authentication" pane in Settings is now "Policy, and has three sections. The "Policy" section is much the same as it was in Mac OS X 10.3. The binding settings are about the same as in Directory Access, however they will override client Directory Access settings. You can also disallow binding if need be. Finally, the "Security" pane allows you to set various security methods. Before you go shutting things off or turning things on here, you'll want to be very careful to make sure that you aren't accidentally killing access for clients that can't use the settings you want. A network that can't be used may be secure, but it's also useless.

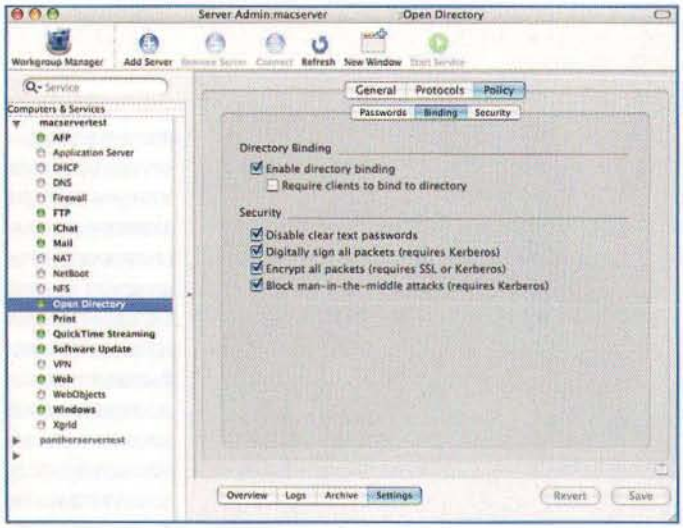


Figure 10: Open Directory Server Policy Settings

One final note: Apple has released a new Open Directory manual that clocks in at over 242 pages. Even if you have been running Open Directory for a while now, read it, take notes, annotate the PDF. A couple days of planning in this manual, and the other manuals available on the Mac OS X 10.4 Server Documentation page can save you months of pain down the line. I also ***highly*** urge you, if possible, to take Apple's server classes, which should already be updated for Mac OS X 10.4. They aren't cheap, but the time you save will more than make up for the cost of the class within months, which is not a bad ROI at all.

Print Service

Considering that Mac OS X 10.3 Server's print service was, to be kind, suboptimal, almost any improvement in Mac OS X 10.4 is welcome. The big news is that instead of ignoring CUPS, (version 1.1.23 in Mac OS X 10.4 Server, from Mac OS X 10.3 Server's 1.1.20) at all but the lowest levels, ala Mac OS X 10.3, Mac OS X 10.4 Server is integrated with CUPs pretty much from to to bottom. This pays off in a number of ways, such as sharing via IPP and better logging, (the CUPS logs are actually useful now). You can set cover sheets in Server Admin, (although custom cover sheets still require some work in CUPS, to create custom sheets and get them in the right spot.).

Another bonus to having the print service better integrated with CUPS is that it makes it easier to use CUPS for the settings that Server Admin or the Apple command line tools don't handle. So, if you want to go beyond what Server Admin gives you, or you want to use different CUPS features, it's a much nicer time to do so.

There's still a couple of annoyances here, such as the only support for authenticated printing in Server Admin is still only via SMB. You can do authenticated printing with CUPS and IPP, but it should be in the Server Admin UI too. Adding shared printers and queues to Open Directory is still a manual operation, which is puzzling. Using Open Directory to manage printers is something that should be automatic. The printer entries should be created as you set them up in Server Admin. Open Directory Printers have to be LPR queues, another annoyance. Bonjour printing is LPR only, however, finding IPP printers shared by Mac OS X 10.4 Server from a Mac OS X 10.4 client is pretty braindead, (They show up in the Printer Browser that comes up when you click "Add" in Printer Setup Utility with a connection type of "Shared Printer"), so it's not as big of an annoyance as it could be. While you can't auto-download Windows client print drivers by default, setting this up is pretty straightforward (and I go over it in the Windows Services section), so Mac OS X 10.4 is a much nicer Windows client print server too. The UI in Server Admin hasn't changed noticeably, other than the aforementioned Cover Sheet and IPP additions. Most of the changes here are under the surface.

QuickTime Streaming Service

While there are a ton of changes in QuickTime itself, hence QuickTime 7, the QuickTime Streaming Server tools having changed as much from Mac OS X 10.3. The biggest changes are to reflect new features in QuickTime, such as H.264, HD Streaming and 3GPP features. The Server Admin UI is about the same as in Mac OS X 10.3, as is the QuickTime Streaming Server Web UI.

QTSS Publisher has had some work done on its UI, mostly to support ease of use and user home directory streaming. There is one big change, and it is not only welcome, but fits the workflow of QTSS Publisher perfectly: AppleScript. QTSS Publisher joins Server Admin and the Gateway Setup Assistant as the only server administration applications from Apple with AppleScript implementations, and QTSS Publisher's is by far the best thought-out and most useful. This is a refreshing change from the first generation AppleScript dictionaries seen of late from Apple, (I'm thinking of Keynote and GarageBand in particular), which are mostly read - only, and of not a great amount of real use. The QTSS Publisher dictionary is full-featured and looks to have been created by folks who not only know what AppleScript can do, but have used AppleScript themselves. The terms are explained, and looking at the object model in Script Debugger's browser, it looks clean and neat. Good job folks!

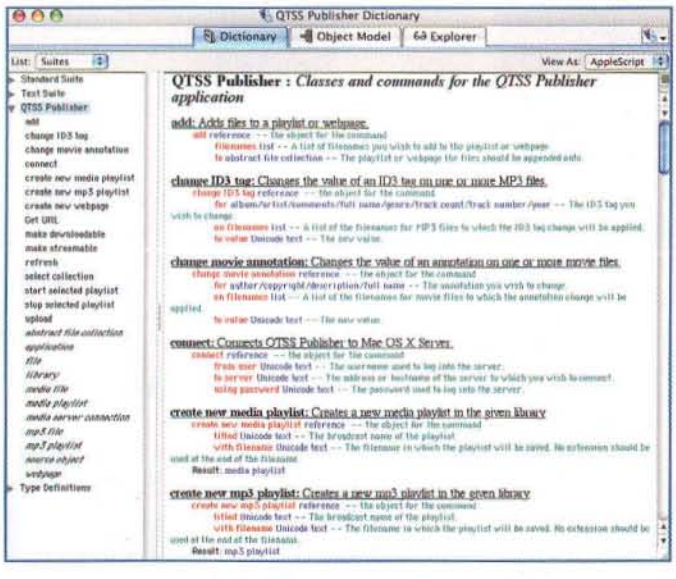


Figure 11: QTSS Publisher AppleScript Dictionary

Software Update Service

The first thing that must be said about the new Software Update service is best quoted from the Mac OS X 10.4 Server documentation:

Note: ***You can't use Software Update service to provide third-party software updates.***

The second thing that must be said about Software Update service, also from the Software Update service documentation:

You can't make your own Software Update packages. For security considerations and to protect attackers from faking packages, the Software Update package installer won't install a package unless it's signed by Apple. In addition, Software Update service will work only with the new package format supported in Mac OS X Server v10.4 or later.

The right or wrong of these items is best left to debate elsewhere, but as of Mac OS X 10.4.0, Software Update service is only for Apple Software Updates, and only for software updates using the new Mac OS X 10.4 package format created by and signed by Apple.

When you connect to Apple's main update servers with your own Software Update service server, Apple does collect the following data from your server:

- Language
- Type
- Browser

So Apple's not doing any deep, evil data mining on your Software Update service server. Having said that, Software Update service is not useless by a long shot. If you have 100 machines, or even ten machines, a way to ensure that your clients only get the updates you approve and don't all have to go out on the Internet to get them is A Good Thing. Yes, you can do this with Apple Remote Desktop, but that's a manual process, and doesn't allow you to assign Software Update service servers to specific clients. It also allows you to better compensate for revoked updates, that Apple may pull for a variety of reasons, since those packages are not presented to users. So, while Software Update service isn't everything, it's better than nothing.

Clients must be running Mac OS X 10.4, which is a bit of annoyance for system administrators who were hoping to use this without having to upgrade their clients to Mac OS X 10.4. As well, you'll want to think about capacity planning well, since pushing out a 50MB update to 500 clients on a 100MB line will make your network cry. If you have a large number of clients, you'll want to consider multiple Software Update service servers, and use Workgroup Manager to assign groups of clients to specific servers.

Looking at Software Update service in Server Admin, you have the Overview Tab, which allows you to see the current status of Software Update service, such as last check, the number of updates that are ***Mirrored***or copied and stored locally for clients, and the number of updates that are ***Enabled***, or made available to clients. You also get the status of the Auto-Mirror and Auto-Enable functions.The Log tab shows you the current contents of the Software Update service log.

The Settings tab has two parts. The first part, "General" allows you to enable or disable automatic mirroring and automatic enabling. Note that the auto-enable is a binary setting. It's on or off, there's no "only on for some updates". You can limit bandwidth between the Software Update service server

and client Macs, and you specify the port you're going to provide the updates on here as well. The "Updates" tab shows you the updates you are currently mirroring and their status.

I will agree that Software Update service is not a complete, full solution for Enterprise IT needs by a long shot, but it can help you manage Apple updates a lot better than was previously possible. That's a win, even if it isn't everything it should be.

VPN Service

This has not changed by huge amounts from Mac OS X 10.3 Server. In Mac OS X 10.4's Server Admin, there are new options for LT2P, such as being able to user MS-CHAPv2 or Kerberos Authentication, and you set the IPSec authentication options for L2TP to either shared secret or SSL Certificates. PPTP is unchanged in Server Admin. Logging and Client Information options remain unchanged from Mac OS X 10.3 Server.

Web Service

Unsurprisingly, the web services in Mac OS X 10.4 Server have their share of updates, the most obvious one being the new Weblog functionality, based on Blojsom, the java - based weblog software. The weblog service gives you fire and forget weblog setup, however, if you're looking for something that gives you point & click functionality on a par with a standard Blojsom install, or MovableType, you'll not be terribly happy with Mac OS X 10.4's implementation. But, if you're looking for a weblog that would be good for smaller children, or you want a simple, basic weblog, then Mac OS X 10.4's weblog is great for that. One thing that is important to note if you plan on hosting multiple sites from a single server: If you turn on weblogs, they're on for ***every site on your server*** so be careful there. Finally, using the Weblog service with clients like ecto is far more tedious to set up than it should be.

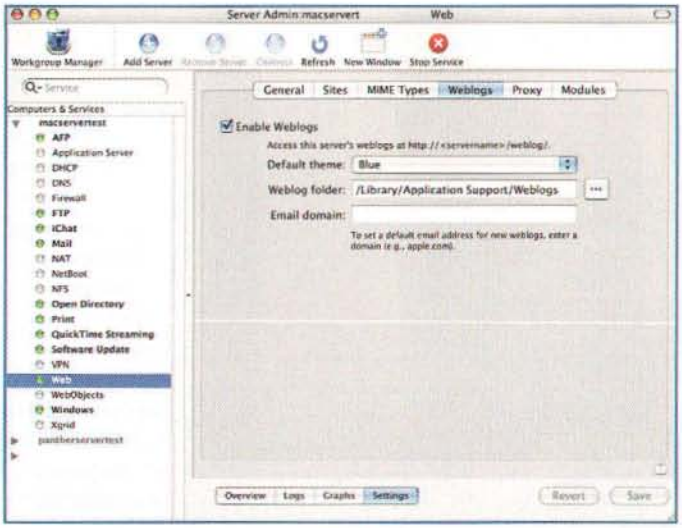


Figure 12: Weblog Settings

There are new authentication options for Apache, most notably Kerberos via Microsoft's SPNEGO protocol. So you can now user Kerberos authentication for not just web pages, but for realms as well. SSL support in Mac OS X 10.4 Server supports SSLv2, SSLv3, and TLSv1, although not every other application is guaranteed to support these versions. The primary version of Apache in Server is at 1.3.33. Apache 2 is provided for testing, in /opt/apache2/. Like every other service on Mac OS X 10.4 Server, Web Services support SACLS.

As we noted earlier, SSL certificate handling is better at all levels, including web services. The way that Apple has made using SSL in the various services in Mac OS X 10.4 Server will make the use of more secure websites much more prevalent.

The performance cache is still in Mac OS X 10.4 Server, but has changed its behavior somewhat, in that you no longer have to keep port 16080 open, since Apache in Mac OS X 10.4 Server handles the performance cache a little more gracefully. If you need to use SSL or Keep-Alive, you can't use the performance cache, and any request containing cookie headers will not have its responses cached. You'll also want to disable the cache for any sites with WebDAV enabled; as it can conflict with uploads.

You can set up search pages in Mac OS X 10.4 Server that can talk to Spotlight. While Spotlight is disabled by default on Server, you can turn it on by changing the Spotlight line in /etc/hostconfig from SPOTLIGHT=-NO- to SPOTLIGHT=-YES-, then start the Spotlight metadata process with **sudo SystemStarter start "MetaData Search"**, or restart the Server. From there, you have to set up the searches for the site/pages, but this is all explained in the Web Technologies Admin Guide from Apple.

WebMail in Mac OS X 10.4 Server is still SquirrelMail, version 1.4.1.

Server Admin hasn't changed much with Mac OS X 10.4 Server, mostly just to handle some of the new features. The "General" area in the Settings now has controls for Allow Persistent Connections, so you can set the number of persistent connections and their timeouts. The "Weblogs" area has all the settings for the Weblog service, which consist of: Enable/Disable Weblogs, set the Default Theme, set the Weblog folder, and the Email domain for emails generated by the Weblog(s). Like I said, it's there, and it's easy. Sophisticated it ain't.

The "Sites" area has had the most changes overall. The Options pane now has a specific Server-Side Include (SSI) setting, and the order has changed. As I said earlier, realms can now use Kerberos authentication, which also requires SSL, and this is in Realms pane. The Security pane has changed to reflect the new SSL integration, like everything else in Server Admin. Finally, the Aliases pane now has different settings for Web Server aliases vs. URL Aliases and Redirects.

WebObjects Service

The WebObjects service is new in Mac OS X 10.4 Server, although WebObjects is hardly new to Mac OS X. Since all this service does is control the WebObjects application server, there's not much to do with it in Server Admin, nor would there be, since WebObjects is more the realm of Xcode. You can get an overview of what's going on with WebObjects if it's running, and you can

set the ports that the WebObjects Monitor, and the wotaskd process run on. You can also enable/disable the monitor.

Windows Service

In a stunning surprise, nothing has changed for Windows services...no, I'm lying, but I'm also running out of ways to say "There's been a lot of changes to <service>. But, once again, there's a lot of new features in the Windows Services in Mac OS X 10.4, and they're stuff that I missed dearly in Mac OS X 10.3.

For sysadmins who aren't using Active Directory, but want to migrate their domains off of Windows NT 4, Mac OS X 10.4 Server supports Backup Domain Controller, (BDC) functionality along with the Primary Domain Controller, (PDC) functions in Mac OS X 10.3 Server. This requires you, (somewhat obviously) to set up the PDC Mac as an Open Directory Master, and the BDC Mac(s) as Open Directory Replicas. All the information is then stored in Open Directory's LDAP store. One thing to note here, since this is for NT4 domains, not Active Directory realms, you have to deal with NetBIOS name restrictions for your network. Machine names should be no more than 15 characters, no special characters or punctuation. This is also a good policy for SMB shares in an NT4 domain.

Mac OS X 10.4 Server is now able to be an Active Directory member server with greater ease and better reliability than before. As I mentioned in the Open Directory section, the setup for this is much simpler and more reliable than in Mac OS X 10.3 Server, and thanks to the ACL support in Mac OS X 10.4 Server, dealing with permissions is far simpler. Thanks to Apple using Windows - compatible semantics in Mac OS X 10.4 Server, you can change ACLs from a Windows box, a plus for people managing AD networks with Macs. Thanks to ACLs, you can finally have nested Windows or Mac groups with access to a share or service, and not run into Unix permissions problems, or the 16 - group limit that Unix permissions forced on you. As with the other services in Mac OS X 10.4 Server, you can also use SACLs to restrict access to the service itself.

Mac OS X 10.4 Server's file locking is improved, so if you enable strict locking for any SMB shares, you should be able to avoid some of the file locking issues that were a problem in Mac OS X 10.3 Server when you had people hitting the same file with different protocols. While Mac OS X 10.4 Server does support oplocks for SMB shares, you should only use those if Windows clients are the ***only*** machines that will be using that share.

On the authentication side, Mac OS X 10.4 Server now supports NTLMv2 and Kerberos for Windows clients. It also supports using Kerberos for Macs accessing other Windows shares, so Mac OS X 10.4 users get Single-Signon for SMB too. As we'll see, you can force NTLMv2 and Kerberos authentication for clients connecting to Mac OS X 10.4 Server, but you should be careful of doing that if you have older Windows boxes, since you have to be running Windows 98 or greater to use NTLMv2 & Kerberos.

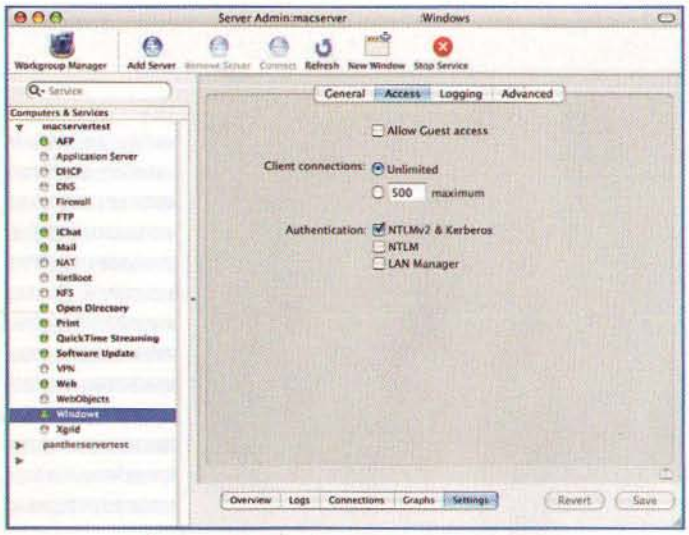


Figure 13: Windows Access Settings

One benefit to all of this is that using Mac OS X 10.4 Server to host ***roaming profiles***, the Windows version of Mac OS X 10.4 Server's mobile home directories is ***much*** easier than it was in Mac OS X 10.3 Server. One caveat with roaming profiles: If you're going to use them for Windows clients, you have to be rather hard-nosed about profile size quotas. With Windows, a Roaming Profile downloads to the client on login, and uploads from the client at logout. If you have a large profile size, (mine is about a GB and a half or so), login and logout on anything but a very fast network can take a long, long, long, time. Enough to where people will think something crashed. Watch your sizes on roaming profiles.

As before with Mac OS X 10.3 Server, Mac OS X 10.4 Server is able to act as a print server for Windows boxes, and can even provide automatic driver download for Windows clients, so they don't have to worry about installing drivers when printing to a Mac OS X 10.4 Server print queue. While the basic documentation is available on the Samba site, as it turns out, doing this in Mac OS X 10.4 Server is a good deal simpler than the Samba howtos show:

1. Add the printer you want shared to Mac OS X Server via Printer Setup Utility

2. Share it out in Server Admin via IPP and SMB (Win2000 and later can use IPP too, but SMB is much easier to use here for a number of reasons. IPP is just better for your Mac clients, but if you won't have any needing to get to this printer, then you can just use SMB)

3. In Workgroup Manager, (WGM) create a share with an SMB name of print$. It must have that exact name as the share name in WGM for SMB sharing, as this is hardcoded into Windows 2000/XP for this function. Make sure you set up the proper ACLs for it, at least giving Domain and Enterprise Admins full control. This is one case where you really want to be one with ACLs, it will make life much easier for windows users and administrators.

4. Create the following subdirectories in print$ in WGM

    a. W32X86 for NT/2000/XP
    b. WIN40 for 9X/ME (Note: The procedure I'm detailing only works for Windows NT/2000/XP clients. You can set it up for 9X clients, but it's much more tedious, and those machines are a bigger hindrance than a help anyway)

5. From a Windows 2000/XP machine, navigate to the "Printers and Faxes" share on the Mac server, and select "Properties" on the printer you want

    a. When asked to install the driver select "No"
    b. In the Advanced tab, click "New Driver..."
    c. Run the Add Printer Driver Wizard (note, if run from an XP machine, this only installs drivers for NT/2000/XP, NOT 9X)
    d. The drivers actually install into the "3" subdirectory in the W32X86 directory (There's a long explanation of why, has to do with earlier versions of NT print drivers living in the kernel. In Windows 2000 and later they aren't. This is all handled automatically for you.)
    e. Configure the standard settings on the printer. Note that device settings will be pre-set this way, but not print job settings, even though you can set them here.

6. Add the printers to the clients either via the printer wizard, or just connecting to them in "Network Places".

That should do it. You can also publish OS X Server's printers in Active Directory via the Active Directory tools, a good idea, since it allows you to just search AD for them. ***(AD's printer search is pretty weak, but that's MS's problem)*** One reason why I recommend ACLs here is that they're a better way to lock down access to a printer, and you can do this from AD tools, which are still a lot nicer to deal with than OD tools. As with computer and share names, watch the 15 - character limit for print share names.

As far as Server Admin goes, there aren't a lot changes. The Overview tab no longer shows you the logging status. In the Settings tab, the "General" panel

adds an option to set up your server as a BDC, and shows you the name of the Active Directory realm if you're joined to one. The "Settings" pane now has a selector for Authentication options: NTLMv2 & Kerberos, NTLM, and LAN Manager. You can select these in any combination you need.

Xgrid Service

Xgrid is Apple's implementation of Grid Computing. Grid computing is a lot of things to a lot of people, but at it's heart, it's just a way to split a task or group of tasks up between multiple computers. Xgrid is not, however, a magic "Make it faster" spell. The job has to be designed for multiple computers to work on. 3-D Rendering, ala Pixar is an example of a job type that can benefit from Xgrid. Burning a DVD is an example of a job type that won't benefit from Xgrid. Grid computing is very big in the scientific community, where jobs are easily set up to benefit from Xgrid or similar implementations. (SETI@Home is an early grid computing implementation.)

Within Xgrid, you have a controller, a client, and an agent. The client submits the task to the controller, which then parcels out the job to a grid, which contains multiple agents. (Yes, you probably could have a single-agent grid, but what's the point?). Apple has two pieces of software that ship with Mac OS X 10.4 Server to help you with Xgrid. The first is the Xgrid Admin utility, the management utility for your grids, which allows you to set up a machine as a controller, and control which grids run which jobs on which agents. Note that while you can have multiple grids run by a single controller, you can only have an agent in a single grid at a time, and a job can only run on one grid at a time. You also can't move an agent between grids while it's running a job, nor can you move a running job between grids.

To avoid problems with rogue agents, there are three levels of security in Xgrid. The highest is Kerberos authentication, which uses Open Directory's Single-Signon to handle authentication. Then there's Password, where you set a single password on the agents, and then set up the controllers and the clients with that same password. There's also no security at all, but that's a bad idea for many reasons. Many, many, reasons.

Within Server Admin, the controls for the Xgrid Service are pretty simple. The Agent Tab allows you to set up the server as an agent, decide how you want to pick your controller, and which controller to connect to if you're picking a specific controller. You can also set how you accept tasks, and what type of controller authentication you want to use. The Controller tab lets you set up that server as a controller, and set up the client and agent authentication types you want to use.

You can have a single server acting as an agent and a controller, (and a client too, for that matter), but it's not a great way to get maximum performance. Note that job submission is done by the **xgrid** command line utility, but if you're using Xgrid, the command line is not going to be anything to worry about.

Conclusion

If you think this is it, well, you're wrong, but unless you want this issue of MacTech to be the "John C. Welch Memorial Issue", we'll end this month's column here. Next month, we'll take a look at Mac OS X 10.4 Server from the Workgroup Manager POV.

Bibliography and References

Almost everything in this article can be found in Apple's Server Documentation, at http://www.apple.com/server /documentation/. What little isn't there, I pried from the ridiculously busy brains of people like Schoun Regan of I.T. Instruction, Michael Bartosh of 4am Media, and Joel Rennich of AFP548.com. Schoun and Michael are the authors of the two best books on Mac OS X Server available, the Visual Quickstart Guide to Mac OS X Server, and Essential Mac OS X Server Administration, respectively. Buy them both, they're great books. If you read any of my columns and *don't* regularly read AFP548.com, then you're missing out on a fantastic resource. All three of these guys, Schoun, Michael, and Joel are Apple Trainers too, a great reason to take the Apple courses if you haven't yet. Those courses are taught by some of the best folks in the Mac market, and well worth the cost.

MT

About The Author

John Welch <jwelch@bynkii.com> is an IT Staff Member for Kansas City Life Insurance, a Technical Strategist for Provar, (http://www.provar.com/) and the Chief Know-It-All for TackyShirt, (http://www.tackyshirt.com/. He has over fifteen years of experience at making Macs work with other computer systems. John specializes in figuring out ways in which to make the Mac do what nobody thinks it can, showing that the Mac is a superior administrative platform, and teaching others how to use it in interesting, if sometimes frightening ways. He also does things that don't involve computertry on occasion, or at least that's the rumor.

What's New In REALbasic 2005?

A look at what's new in REAL Software's recently released update to REALbasic.

By Will Leshner

REALbasic 2005 Is Out

REAL Software recently released an all new version of REALbasic called REALbasic 2005, and you might be wondering what all the fuss is about. REALbasic 2005 was first announced at REAL World (REAL Software's annual REALbasic developers conference) in 2004, and it was finally released in June of this year. It sports an entirely redesigned user interface, new controls, additions to the REALbasic language itself, and a new "Rapid Release" shipping schedule that promises a new version of REALbasic every 90 days.

To really appreciate some of the things I'll be talking about, you'll want to run REALbasic for yourself. If you don't have a license, you can download the REALbasic demo from the REAL Software website.

New IDE

Made With REALbasic

With the release of REALbasic 2005, REAL Software hasn't just redesigned the REALbasic IDE, they have also rewritten it in REALbasic. The IDE itself is now the best example of a made with REALbasic application. This alone is a great step forward for REALbasic, because it means that the REAL Software engineers are now fulltime REALbasic developer, and they will now benefit from the improved efficiency that working in REALbasic provides. That means features should be quicker to implement and should appear in releases more quickly. It also means that the REAL Software engineers will experience the same bugs and design flaws in REALbasic as their customers, making it much more likely that fixes will appear sooner, rather than later.

All-In-One Window

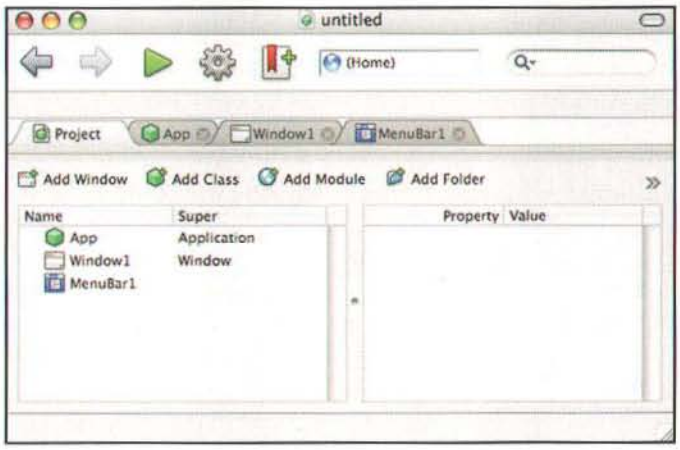


Figure 1. All-in-one project window.

The most obvious change to the REALbasic IDE is the all-in-one project window (as shown in Figure 1). This change may take some getting used to, but it has several advantages over the old multi-window approach. First, the new IDE is a joy to use on Windows. If you haven't had a chance to use older versions of REALbasic on Windows, you may not be aware of some of the problems, but there were several and they affected productivity substantially. One problem with the old IDE was that it used floating windows for both the controls palette and the properties editor. While a floating window approach worked

well on the Mac, it didn't work well on Windows, where the floating windows would fight with the main IDE window for the focus. The main window itself was also a problem on Windows. It was an MDI-style window that severely limited a developer's ability to switch from one code or layout window to another.

Another feature of the new IDE's one-window approach is that it simplifies having more than one project open at the same time. A serious limitation with the old IDE was its inability to open more than one project at a time. Developers managed to work around that limitation through tricks, such as having more than one copy of the REALbasic application running at the same time. With REALbasic 2005, such tricks become unnecessary, because opening multiple projects is fully supported by the IDE. Furthermore, having multiple projects open at the same time is easy to manage for developers because each project is fully contained within its own window.

IDE As Web Browser

The REALbasic 2005 IDE models itself on a tabbed web browser. In a web browser, you visit web pages. In the REALbasic 2005 IDE, you visit editors. Each editor you visit in a project can be thought of as a location, just as the different web pages you visit in a web browser can be thought of as locations.. In the REALbasic 2005 IDE, you can jump directly to a location using the Location field at the top of every project window. For example, if you wanted to jump directly to the Open event of the App object, you would type App.Open in the Location field and hit return to have the IDE take you directly to the code editor for that event. Also like a web browser, as you visit code editor locations, the IDE will remember where you've been. You can move back and forward through the locations you have visited by clicking the backward and forward buttons in the IDE's main toolbar (see Figure 2).

Figure 2. Browser-like project navigation controls.

If you find that there is a particular location you visit a lot, you can set a bookmark to that location. Choosing a bookmark from the Bookmarks menu opens the location for that bookmark. Bookmarks are global across all projects, which means that you can select a bookmark for a location in a project that isn't open, and the IDE will automatically open the project and take you directly to the location in that project. There is also a favorites bar, located below the main toolbar, into which you can place bookmarks that you use a lot. The bookmarks in the bookmarks bar are local to a project and will not show up in other projects.

Tabbed Editors

Most of your time using the REALbasic IDE will likely be spent in the main content area, located below the main toolbar and the favorites bar. The main content area is tabbed, with each location you visit contained within its own tab (see Figure 3). One of the advantages to the new tabbed interface is the ability to quickly switch from one location in a project to another simply by clicking tabs. If you need to see two locations at the same time, it is possible to open another project window for the current project by choosing New Window from the File menu.

Figure 3. Project tabs.

Most tab locations have the same basic structure. At the top is a bar of buttons for common actions specific to that particular location. The Project tab, for example, has buttons for adding classes, modules, and menu bars, to a project. A code editor tab has buttons for adding methods, properties, events, and other code elements (Figure 4 shows the menu editor toolbar). Below the command bar is the main content area for the tab. The look of the main content area will depend on the type of tab. The Project tab contains a list of all the classes, windows, modules, and other elements contained within that project. A code editor tab contains a list of all of the methods, properties, events, and other code elements for the item being edited, together with a code editor pane for editing those items. A window layout tab contains a pasteboard for editing a window, a list of controls to add to the window, and a properties editor for editing the properties of the window or the items contained on that window.

Figure 4. Menu editor command bar.

Redesigned Editors

Most of the editors in REALbasic 2005 have been redesigned, and perhaps the biggest change is the way the new window editor works. The old IDE's window editor looked, and in many ways acted, like a real window. There were, however, several problems with editing a window that itself acted like a real window. First, every class instantiated directly on the window had to be contained within the window itself. Some classes, like Timers and TCPSockets, are not really controls and do not appear in the UI of the running application. The presence of these non-controls during the design phase tended to clutter the window, and generally got in the way when trying to position and style those elements that really were controls. Another problem with the old window editor was that you could never design a window that was larger than your monitor. If you had a small monitor and a large window, you were forced to play games with the position of the window in order to design it. Lastly, there was a problem with the old window editor on Windows that had to do with maximizing all the windows in the project. Because a window editor was just another window as far as Windows is concerned, it would get

maximized along with all the other windows, which was clearly not the desired behavior.

The new window editor in REALbasic 2005 fixes many of the problems of the old window editor. The window being edited is now contained within a pasteboard, and you can drop objects anywhere on the pasteboard, not just on the window. So, for example, if a window is to have a Timer on it, the Timer can be placed on the pasteboard, next to the window, keeping it outside the confines of the window itself. Also, the pasteboard has scrollbars, which means that you can edit a window of any size, regardless of the size of your monitor. Finally, the new window editor solves the problem on Windows where the window being edited is treated like a real window. The window is actually just a picture in the IDE, so Windows won't maximize it in unfortunate ways.

In addition to the window editor, the code editor has also gotten a face lift in the new IDE. The most obvious change is the addition of guidelines to show you which code blocks belong together. The guidelines are especially useful when the code in a method has several nested blocks and the code itself is a bit too long to all fit on one screen. The lines make it immediately obvious which End If goes with which If. The lines also server a functional purpose. Clicking a line selects all the text in that block.

Most Improved: Searching

Perhaps one of the best features of the new REALbasic IDE is its vastly improved searching options. There is still a traditional find dialog, from which you can find and replace text throughout a project. New, in the REALbasic 2005 IDE, however, is the ability to perform regular expression and whole word searches. Search really shines, however, when you do a Find All. All found items are displayed in a separate search tab. Double-clicking an item in a search tab brings you directly to the location for that item. But you don't have to leave the search tab in order to replace some or all of the found items with new text. At the bottom of each search tab is a field into which you can type replacement text. To perform a replace, all you have to do is type replacement text, select the items in the list of found items you want to replace, and click the Replace button.

You do not need to visit the Find dialog in order to search for items in a project. There is a search field in the main toolbar of the project window into which you can type text to do an instant search. A popup in the search field lets you set the scope of the search to the entire project, the current item, or the current method. If you are running Mac OS X Tiger, then you get one more option that is really cool: computer. A computer search uses Apple's new Spotlight technology to search all the REALbasic projects on your computer for a given search term. In order to use this option, you will need to download and install REAL Software's Spotlight plugin for REALbasic. Once you do, you will be able to perform lightning-fast searches of all of the projects on your Mac. As with any search, the results will appear in their own search tab. Double-clicking one will automatically open the project for that item, if the project is not currently open.

Customizable IDE

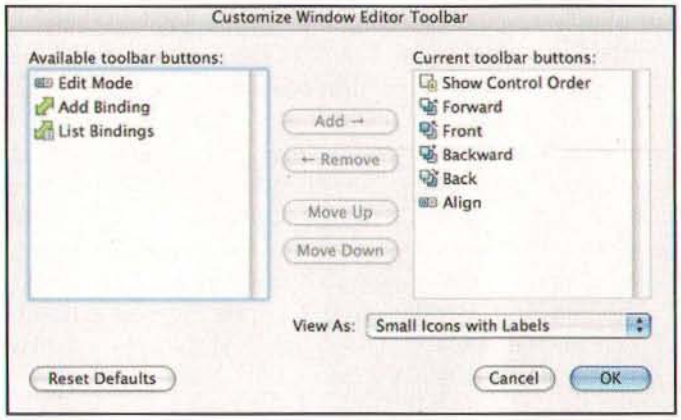


Figure 5. A customize dialog.

The new REALbasic IDE is much more customizable that the previous versions. For example, every toolbar you see can be customized. If you don't like a particular command button, you can remove it. If you don't like the order of the buttons, you can rearrange them. Some command buttons aren't even visible by default. Add Binding and List Bindings, for example, are two commands you might like to have easy access to that also happen to have buttons associated with them. By default, window editor toolbars don't include those to buttons. To add them, just bring up the customize dialog for the window editor toolbar (shown in Figure 5) and add either or both of them.

The only downside to REALbasic 2005's toolbar customization process is that you need to be looking at a toolbar in order to edit it. It may not occur to you that some toolbars can be edited. The debugger toolbar, for example, lacks a Run button, but you can add a Run button using the debugger's toolbar editor. You must be running in the debugger in order to access that editor, however.

Much More

I've only scratched the surface of all of the new features to be found in the REALbasic 2005 IDE. In addition to those things I've already talked about, there is also a totally new Language Reference, a new debugger, file type sets, and code refactoring tools. I'll leave these new features for you to discover on your own.

New Language Additions

The IDE isn't the only thing to get new additions in REALbasic. The REALbasic language has had a few additions to it as well, and all of them are welcome. Here are a few of the more exciting additions.

Declaring Variables

It was once the case that REALbasic required all variable declarations to appear at the top of a method, before any other

code. That restriction was relaxed in REALbasic 5 so that a Dim statement could appear anywhere at the top level of a method. Now, in REALbasic 2005, variables can be declared inside blocks as well. And, as you would expect, the scope of such a variable is the block itself. Which means that the following silly code, while illegal in REALbasic 5, is perfectly legal in REALbasic 2005:

```
dim x as Integer = 2
  if x = 3 then
    dim y as Integer = 4
  x = y
end if
```

Another addition to the REALbasic language having to do with declaring variables is the ability to declare the loop variable of a for statement directly in the statement itself, as this code demonstrates:

```
for i as Integer = 0 to 10
  j = j + i
next
```

New Static Keyword

One thing sorely lacking in REALbasic is class properties and class methods. As in C++, static properties and methods would be shared by all the instances of a class. They enable class instances to coordinate their behavior, and they also make the Singleton design pattern a possibility (you can fake singletons now in REALbasic, but you can't get the compiler to enforce them for you). REALbasic 2005 doesn't introduce static property and class methods, but it does introduce Static as a keyword of the language and that gets us half way to static properties and methods.

Static is used like Dim to declare variables in a method. The difference is that variables declared Static persist across calls of the method, and their value is shared by all callers of that method (in other words, Static works in REALbasic the way it works in C). Static variables can be declared in module method or class method. Static variables in class method are static to the method and not to any particular instance of that method. Which means that every instance of the method sees the same static variable.

Static variables declared in methods of a class can be used by instances of that class to communicate with each other. One common example is keeping running total of all of the instances of a class that are currently in existence. Things are a little tricky because the static variable is private to the method in which it is declared, so we can't really define a cluster of support methods around one static variable. Instead, what we can do is create one method to which we pass commands for acting on the static variable. In the case of an instance count, we would need commands for incrementing the count, decrementing the count, and returning the count. Our final method might look something like the following:

```
private function InstanceCount(aCmd as String) as Integer
  static  count as Integer
  dim result as Integer

  select case aCmd
  case "INCREMENT"
    count = count + 1
  case "DECREMENT"
    count = count - 1
  case "GET"
    result = count
  end select

  return result
end function
```

To use InstanceCount, we would simply need to call InstanceCount("INCREMEMNT") in the constructor for our class, and InstanceCount("DECREMENT") in its destructor. We could also provide a public method for getting the instance count that would just return InstanceCount("GET").

Soft Declares

Another very useful new feature in REALbasic is the ability to create soft declares. Declares are a way to call a functions in shared libraries. They are commonly used to get at platform-specific functions of the operating system that are otherwise unavailable to pure REALbasic code.

Up until REALbasic 2005, a function specified in a declare was hard-coded in the executable and resolved when the application launched. If the OS couldn't figure out what shared library and function a particular declare referred to, then the application would fail to launch. In REALbasic 2005, it is now possible to make a declare "soft", which means that the declared function is not resolved until runtime. A new IsFunctionAvailable method has also been added to the System object. IsFunctionAvailable makes it possible to determine, at runtime, whether a function specified in a declare is or is not available, and take action accordingly.

An example may help to clarify the issue. I'll use a silly declare to make the point, but I hope you can see how the idea can be extended to other, more interesting declares. Imagine we are writing a custom control and we need to know the system's double-click time in order to handle double-clicks properly. There is a Carbon function called GetDblTime that will return that number, so we dutifully construct the following declare:

```
Declare Function GetDblTime Lib "Carbon" () as Integer
```

At which point we decide to test things out, so we run the project. There are two possible outcomes, depending on whether we are building a Mach–O application or a PEF application. If we are building a Mach–O application, our application will launch and the above declare will perform correctly. If, however, we are building a PEF application, we will get an error dialog at application launch telling us that the Carbon shared library could not be found. That is because the PEF loader doesn't know about a shared library called "Carbon", as we have identified it in the declare above. The PEF loader does know about a shared library called "CarbonLib", and if we had constructed the above declare using "CarbonLib" instead of "Carbon", then our PEF application would launch correctly.

A soft declare solves this problem. It defers the resolution of the "Carbon" shared library to runtime, instead of load time. Furthermore, it is the REALbasic framework that does the

resolving and not the PEF loader, so it can be a bit smarter about handling a declare to "Carbon" in a PEF application. Turning the above declare into a soft declare is as easy as adding the Soft keyword to the front:

```
Soft Declare Function GetDblTime Lib "Carbon" () as Integer
```

Amazing as it may seem, simply adding the Soft keyword resolves the entire PEF versus Mach–O problem. The declare will now function correctly in both kinds of applications.

Soft declares solve a whole host of problems. On Linux, for example, there is a particularly nasty problem having to do with declaring to functions in LibC. On many distributions of Linux, LibC is really just a linker script for gcc that tracks down the real LibC shared library. Before REALbasic 2005, it was necessary to construct declares using a path to LibC itself, which resulted in code that was decidedly non-portable. Furthermore, you'd have to know where LibC lived on the particular distribution of Linux on which you were planning to run your application. Soft declares solves this problem as well. In REALbasic 2005 all you have to do is create a soft declare to "LibC" and then REALbasic resolves the real location for you at runtime.

New Controls And Classes

In addition to a brand new IDE and additions the REALbasic language, REALbasic 2005 also comes with several new classes and controls that developers can add to their bag of tricks. I'll cover a couple of them here.

HTMLViewer

Figure 6. HTMLViewer on Mac OS.

Figure 7. HTMLViewer on Windows.

REALbasic developers have long wished for a built-in HTML control and in REALbasic 2005 that wish has been granted. The HTMLViewer uses the default HTML rendering technology on each platform. On Mac OS, that technology is WebKit, on Windows it is Internet Explorer, and on Linux it is Mozilla.

I was lucky enough to be attending WWDC when Apple first announced WebKit. I really enjoyed the web-browser-in-three-steps demonstration that the Apple engineers performed during several of the sessions I attended. Now, that same demonstration is possible with REALbasic, and the result is a web browser that runs on Mac OS, Windows, and Linux. It doesn't take much to whip up a web browser in REALbasic 2005. In fact, it only takes one line of code. To begin, start with a blank desktop project. Drop an HTMLViewer control, an EditField, and a PushButton onto Window1. Then, in the Action event for PushButton1, add this code:

```
HTMLViewer1.LoadURL EditField1.Text
```

That's all there is to it. Run the project, enter a URL into the edit field, push the button, and start browsing. Figures 6 and 7 show the resulting web browser running on both Mac OS and Windows.

ContainerControl

Before REALbasic 2005, if you had a set of controls that worked together as a group, and you wanted that same set of controls to appear either in different places on the same window, or in different windows, then you had no choice but to recreate the exact same set of controls in each place that you needed them to appear. That problem is solved in REALbasic 2005 with the introduction of the ContainerControl. A ContainerControl acts a little bit like a window in that you design one in a window editor. But instead of appearing as a separate window, a ContainerControl is attached to another window, either at design time, or in code at runtime. One ContainerControl can be attached to any number of windows. In fact, you can even embed one ContainerControl inside another

ContainerControl. Furthermore, ContainerControls are first-class citizens in REALbasic. You can attach properties, events, and methods to them in order to encapsulate their behavior in the ContainerControl itself.

There are a number of ways that ContainerControls can be used to simplify the creation of a complex user interface. For example, creating resizable panes becomes almost trivial with ContainerControls. By locking the bounds of each control contained within the ContainerControl appropriately, it is a simple matter of resizing the ContainerControl itself to get the effect of a resizable pane.

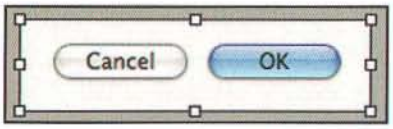


Figure 8. ContainerControl with two buttons.

Let's work through an example to see how ContainerControls can be used to easily solve a tricky problem. Anybody who has had to create cross-platform dialogs is well aware that the order of the OK and Cancel buttons as they appear in a dialog is different on Windows and the Mac. In order to have the buttons appear in the right order on each platform, it is usually necessary to write custom code in each dialog's Open event to take care of switching the buttons to the right order, depending on the platform. With ContainerControls, however, this becomes a simple task that can be solved once and forgotten. Start by creating a new ContainerControl in an empty project. Onto the ContainerControl drop two PushButtons, changing the Caption property of one PushButton to OK, and the other to Cancel. It is necessary to arrange the buttons in the proper order for one of the platforms on which the application will run. In Figure 8 I have arranged the butons in Mac OS order. The next step is to place code in the ContainerControl's Open event to switch the button order when the application is running on Windows (the buttons are already in the right order for the Mac). This requires a #if conditional compilation directive, as follows:

```
#if TargetWin32
  dim bp as Integer
  bp = CancelButton.Left
  CancelButton.Left = OkButton.Left
  OkButton.Left = bp
#endif
```

The final step is to place the ContainerControl inside of another window, where it can be instantiated when the window is created. Figure 9 shows what this might look like. If you now create Windows and Mac OS versions of the app and run them on their respective platforms, you should see the OK and Cancel buttons appear in their correct positions for each platform. Furthermore, the ContainerControl we created to solve this problem is reusable. We can drag it to any window or dialog box to get the exact same effect.

Figure 9. ContainerControl on a window.

BinaryStream

In addition to new controls, REALbasic 2005 comes with several new classes, as well as additions to existing classes. One addition that is particularly welcome are extensions to the BinaryStream. BinaryStreams are used to read and write binary data in a file. That hasn't changed in REALbasic 2005. What has changed is that BinaryStreams can now be used with MemoryBlocks and Strings as well. The trick to getting a BinaryStream to use a MemoryBlock or a String is to pass a an existing String or MemoryBlock to a BinaryStream when constructing one. For example, here is how you would create a BinaryStream that operated on a MemoryBlock, instead of a file:

```
dim mb as MemoryBlock
dim bs as BinaryStream
mb = new MemoryBlock(0)
bs = new BinaryStream(mb)
```

A BinaryStream that has been created with a MemoryBlock can be used just like any other BinaryStream. The difference is that instead of reading and writing data from a file, the BinaryStream will read and write data from a MemoryBlock

instead. In the past, when I needed a memory-based stream class, I wrote it myself. But the new REALbasic implementation is much better than my home-grown one. Because BinaryStream is the stream class in all cases (MemoryBlock, String, or file), code can be written that reads and writes BinaryStreams that never has to know that the underlying data lives in a MemoryBlock instead of a file. That means more general code that can be reused in interesting ways.

REAL SQL Database

One new feature in REALbasic 2005 that is of particular interest to me is the addition of a new single-user database engine based on SQLite. In case you haven't heard of it, SQLite is a powerful and speedy SQL database engine with a growing user base. In fact, Apple has included SQLite in Mac OS 10.4 and provided access to it through its new CoreData technology. By choosing SQLite, REAL Software has guaranteed that REALbasic developers will have access to a state-of-the-art, high-performance database engine.

Access to the SQLite database engine is through the REALSQLDatabase class. The REALDatabase class still exists for backward compatibility, but it has been deprecated, and REAL Software suggests we move to the new database engine as soon as possible. The API for the new REALSQLDatabase class is exactly the same as that of the REALDatabase, so moving code over to the new database should be as easy as finding every occurrence of REALDatabase and replacing it with REALSQLDatabase. It will be necessary to convert any existing databases, however. The REAL SQL Database only understands SQLite databases (and only SQLite 3 databases, if that means anything to you). REAL Software provides a conversion utility on their website to convert databases made with the REAL Database into REAL SQL Databases.

The strength of SQLite is its speed, robustness, and ability to handle a wide range of SQL syntax. If you are looking for a way to have both a local single-user database and a remote multi-user database in the same application, then the REAL SQL Database is definitely worth a look. By taking advantage of REALbasic's Database classes and the database engines that support them, it is possible to talk to several different databases at once in the same application. For example, you could create an application that talked to a MySQL database if it is available, but fell back to a local REAL SQL Database when access to the MySQL database isn't available. For the most part, you could even feed exactly the same SQL to both database engines, making the transition between the two seamless.

New Platform

Before REALbasic 2005 was released, it was possible to run the REALbasic IDE on three platforms: Mac OS Classic, Mac OS X, and Windows. REALbasic 2005 loses one platform, but gains another. The lost platform is Mac OS Classic. While it is still possible to build Classic applications, the REALbasic IDE itself is not supported on that platform. In exchange for Classic, we get Linux. Although it has yet to go final, REAL Software is currently conducting on open beta of the Linux IDE. REAL Software has also announced that the Standard version of the Linux IDE will be free. I have to admit that I'm not as familiar with Linux development as I am with Mac and Windows development, but now that there will be a development IDE that I am already familiar with on Linux, I'm looking forward to adding Linux to the platforms I can support with my software products.

New Release Schedule

As if REALbasic 2005 and all of its exciting new features aren't enough, REAL Software has also announced a new release schedule to go along with it. In the past, new major versions of REALbasic arrived anywhere from six to eighteen months apart from each other. This created a problem for both REAL Software and REALbasic developers. For REAL Software, the problem was that while their engineers were furiously coding the next major release, they were also maintaining the current release. As with all software, REAL Software periodically released maintenance releases of REALbasic in order to provide bug fixes to developers waiting for those fixes. Maintenance releases would contain bug fixes, and occasionally minor features, but never the big new features that were being prepared for the major release. Sometimes a particular bug fix was too risky to ship in a maintenance release, so it, and the developers waiting for it, had to wait for the major release.

REAL Software's new release schedule is based on a rapid release model in which a new version of REALbasic will ship at least every 90 days. New releases will included both bug fixes and new features. With this new rapid release model, there will no longer be a need to maintain one release of REALbasic while working on the next release. Upcoming releases are always both the maintenance release for the current version, as well as the next major release with new features.

With the new rapid release model comes a new way to buy REALbasic. Instead of paying to upgrade REALbasic at every major release, developers will pay once for all the updates in a given period of time. New licenses include six months of updates. Upgraded and renewed licenses include twelve months of updates.

Conclusion

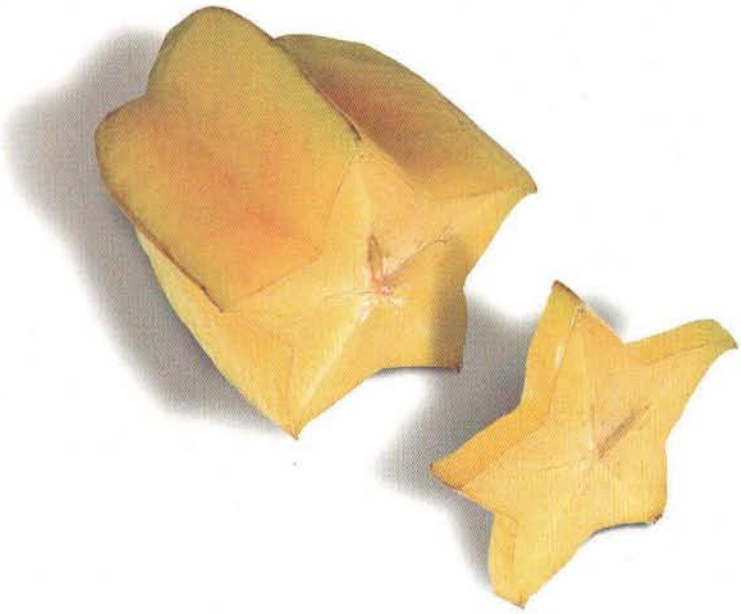
As you can see, there is plenty that is new in REALbasic 2005. The big change, of course, is the new REALbasic IDE, but almost every aspect of REALbasic has something new to explore. If you haven't yet given REALbasic a try, you should. REALbasic has always been a good tool for developers to have in their toolbox. With REALbasic 2005, that tool got even better.

MT

About The Author

Will Leshner is a full-time software engineer who has also been developing software with REALbasic for over five years. He writes the From Scratch column in REALbasic Developer, and he also has a weblog devoted to REALbasic, which can be found at http://rbgazette.com.

Shine.



Apache Bootcamp

Cocoa® Bootcamp

Core Bootcamp

PHP 5 Bootcamp

PostgreSQL Bootcamp

Python® Bootcamp

Core Mac OS® X Bootcamp

Get your dark sunglasses ready. Tell your boss to roll out the red carpet. Are you ready for your breakthrough role? Director Mark Dalrymple, author of *Core Mac OS® X and Unix Programming*, will take you from indies to a blockbuster in five days.

Sample Scenes:

- Multithreading
- Rendezvous™
- System Daemons
- Distributed Objects
- Network Programming
- Authentication & Authorization

Big Nerd Ranch offers intensive training classes for developers and administrators. Your expert instructor guides you through a rigorous week of learning. You leave ready to start developing (but with instructor support if you get stuck).

Big Nerd Ranch: Baby, we'll make you a star!

www.bignerdranch.com • 678.595.6773 • roundup@bignerdranch.com

WORKING WITH TEXT

When writing AppleScript code, many of the things that you will automate will involve working with and manipulating text in some manner. For example, you might need to write a script that will retrieve text content from a FileMaker Pro database, and then place that content into an Adobe InDesign document. You may need to maintain a text-based log file of your script's activity during processing, or you may need a script that will extract content from email messages, and write the content to files on a server.

During this month's article, we will discuss a number of ways to work with text, including ways to break text apart, search text, and read from and write to files.

About Text in AppleScript

Much like a scriptable application in the Mac OS, the AppleScript language itself possesses classes and commands. These classes and commands are considered to be the ***core language*** of AppleScript, and are used, interspersed with application and scripting addition terminology, to make up your scripts. For a detailed overview of AppleScript's core language, you should refer to ***The AppleScript Language Guide***, which is available through Apple's Developer Connection at http://developer.apple.com/documentation/AppleScript/.

In AppleScript, text is considered to be a class, and is synonymous with the class **string**. Because of this, throughout this article, I will use the term **string** when referring to text.

```
class of ("This is some text" as text)
-> string
```

Just like classes in applications, AppleScript core language classes can possess properties. A string possesses a **length** property, which may be used in order to determine the number of characters contained within the string. For example:

```
length of "This is some text"
-> 17
```

Manipulating Text

When working with a string in AppleScript, one of the things that you will probably want to do is to manipulate it, or break it apart in some way. For example, you might need to write code that will parse a tab-delimited file, extracting field information. Once broken apart, it can be repurposed, merged back together in various ways, and more.

Elements of a String

In AppleScript, **paragraphs**, **words**, **characters**, and **text** are all considered to be elements of the class **string**. Because of

this, a string can be broken up in a number of different ways. The following examples show ***some*** of the ways that a string can be broken up by referencing its elements.

The following example code will retrieve a paragraph from a specified string:

```
set theText to "This is paragraph 1 of some text.
This is paragraph 2 of some text."

set theParagraph to paragraph 2 of theText
-> "This is paragraph 2 of some text."
```

The following example code will retrieve a word from the string specified above:

```
word 3 of theText
-> "paragraph"
```

The following example code will retrieve a character from the string specified above:

```
character 9 of theText
-> "p"
```

You may also choose to retrieve multiple elements of a string at once. The following code will retrieve a specified set of characters from the string specified above:

```
characters 1 thru 9 of theText
-> {"T", "h", "i", "s", " ", "i", "s", " ", "p"}
```

When retrieving elements in this manner, you will notice that the result is provided as a list. When retrieving words or paragraphs in this manner, a list may suffice. However, when retrieving characters, you may prefer a string instead. To retrieve a list of characters as a string, you could coerce the retrieved list back to a string. You could also reference the text element of the string, rather than the character element. For example:

```
(characters 1 thru 9 of theText) as string
-> "This is p"

text 1 thru 9 of theText
-> "This is p"
```

Using the Offset Command

At times, you may need to determine the location of a specific character, word, or string within a longer string. While this could be accomplished by using a repeat statement to loop through the characters of the string until the specified search string is found, a more efficient way would be to use the **offset** command. The **offset** command is included in the ***String Commands*** suite in the ***Standard Additions*** scripting addition that is installed with Mac OS X.

```
set theFileName to "filename.jpg"

offset of "." in theFileName
-> 9
```

As you can see from the example code above, the **offset** command will return the position of the first instance of a specified string within another string. With this value, you can then retrieve specific parts of the string. For example, the following sample code will extract the prefix before a specified character in the string that we used above.

```
text 1 thru (offset of "." in theFileName) of
theFileName
-> "filename."
```

Note in the example above, that the extracted prefix actually contains the delimiter character. Again, this is because the **offset** command will return the actual position of the first instance of the specified string. In order to extract the prefix without the delimiter, then you must subtract 1 from the offset. For example:

```
set thePrefix to text 1 thru ((offset of "." in
theFileName) - 1) of theFileName
-> "filename"
```

You may also add 1 to the offset, and extract text from that location until the end of the string, in order to retrieve the suffix following the delimiter. For example:

```
set theSuffix to text ((offset of "." in theFileName)
+ 1) thru -1 of theFileName
-> "jpg"
```

Again, the **offset** command will return the position of only the first instance of a specified string. However, what if a string contains multiple delimiters, and you want to break the text apart based on the offset of the last delimiter? To do this, you can extract the characters of the string in list format, then reverse them using the reverse property of a list. Next, you can change the reversed characters back to a string, extract the prefix and suffix, and then reverse them back. This sounds complicated, but it can actually be done in only a few lines of code. The following example code will walk you through the process.

This example code will extract the characters of the string:

```
set theFileName to "file.name.jpg"
set theCharacters to characters of theFileName
-> {"f", "i", "l", "e", ".", "n", "a", "m", "e", ".",
"j", "p", "g"}
```

This example code will reverse the extracted characters:

```
set theReversedCharacters to reverse of theCharacters
-> {"g", "p", "j", ".", "e", "m", "a", "n", ".", "e",
"l", "i", "f"}
```

This example code will convert the reversed characters back to a string:

```
set theReversedFileName to theReversedCharacters as string
-> "gpj.eman.elif"
```

This example code will locate the delimiter in the reversed string, using the **offset** command:

```
set theOffset to offset of "." in theReversedFileName
-> 4
```

This example code will retrieve the prefix and suffix from the reversed string:

```
set theReversedSuffix to text 1 thru (theOffset - 1) of
theReversedFileName
-> "gpj"

set theReversedPrefix to text (theOffset + 1) thru -1 of
theReversedFileName
-> "eman.elif"
```

This example code will reverse the extracted prefix and suffix back to their original form:

```
set thePrefix to (reverse of (characters of
theReversedPrefix)) as string
-> "file.name"

set theSuffix to (reverse of (characters of
theReversedSuffix)) as string
-> "jpg"
```

Now, you should have the properly retrieved prefix and suffix. The example code above could actually have been written in a more condensed fashion. It was intentionally written in a verbose manner for demonstration purposes. For example, the following code will perform the same function, but has been condensed into fewer lines of code:

```
set theFileName to "file.name.jpg"
set theReversedFileName to (reverse of (characters of
theFileName)) as string
set theOffset to offset of "." in theReversedFileName
set thePrefix to (reverse of (characters (theOffset + 1) thru
-1 of theReversedFileName)) as string
set theSuffix to (reverse of (characters 1 thru (theOffset -
1) of theReversedFileName)) as string
```

Another thing to note when working with the **offset** command is that in some cases, you may attempt to get the offset of a string that does not exist with the string you are evaluating. If this occurs, the **offset** command will return a value of **0**. For example:

```
offset of "." in "filename"
-> 0
```

As you begin using the **offset** command, be sure to add code to handle this type of situation, should it occur.

Using AppleScript's Text Item Delimiters

Another way of breaking text apart is by making use of **AppleScript's text item delimiters** property, which is actually a property of AppleScript itself, and can be retrieved or changed at any time. **AppleScript's text item delimiters** property contains the delimiter that is used to separate chunks of text within a string. By default, **AppleScript's text item delimiters** property is set to a value of {""}, essentially an empty string.

> Though AppleScript's text item delimiters may be set to a list containing multiple values, AppleScript will only utilize the first value in the list. For this reason, when setting AppleScript's text item delimiters, it is not necessary to specify a list. Rather, a string may be used, as you will see in the next code example.

```
AppleScript's text item delimiters
-> {""}
```

A character is the smallest element within a string. Since AppleScript's text item delimiters are set to an empty string by default, retrieving the text elements from a string will return the characters from within that string in list format.

The following example code will demonstrate how AppleScript's text item delimiters may be changed in order to break apart a string. Please note that modifying this property of AppleScript may affect other code in your script. Therefore, you should always be sure to set the value of the property back to its default value when you are done manipulating your string.

```
set theText to "01.01.2005"
set AppleScript's text item delimiters to "."
set theTextItems to text items of theText
set AppleScript's text item delimiters to {""}
theTextItems
-> {"01", "01", "2005"}
```

As you can see, the example code above can be used to convert a string to a list, using a specified delimiter. So, using this method, you could easily write code that would convert a tab delimited string into a list of fields.

The **AppleScript's text item delimiters** property may also be used to coerce a list of values back to a string. The following example code will take the list output by the previous example, and change it back to a string, using a different delimiter.

```
set theTextItems to {"01", "01", "2005"}
set AppleScript's text item delimiters to "-"
set theText to theTextItems as string
set AppleScript's text item delimiters to {""}
theText
-> "01-01-2005"
```

Now that we have explored ways to convert a string to a list and back, we can take things a step further. The following example code will perform a find and replace within a string.

```
set theText to "01-01-2005"
set AppleScript's text item delimiters to "-"
set theTextItems to text items of theText
set AppleScript's text item delimiters to "/"
set theText to theTextItems as string
set AppleScript's text item delimiters to {""}
theText
-> "01/01/2005"
```

In the example code above, every instance of the "-" character is replaced with the "/" character.

In all of the examples above, we were working with a single character as our delimiter. If desired, you may set AppleScript's text item delimiters to a longer string containing multiple characters, such as a word or a paragraph.

Reading and Writing Text

Now that we have explored several ways to break apart and manipulate text, let's discuss ways to work with files through reading and writing.

Reading from a File

Reading from a file is done using a command found in the ***File Read/Write*** suite of the ***Standard Additions*** scripting addition. To read from a file, use the read command.

```
set theFile to choose file with prompt "Select a text file:"
read theFile
```

The example code above will prompt you to select a text file. Next, it will read the file and return the entire contents of the file as a string.

When reading from a file, you may optionally choose to use the open for access command, also found in the ***File Read/Write*** suite, to open a file, prior to reading from it. By using this command to open a file prior to reading from it, the file will remain opened in memory until the script closes the file, using the close access command. For example:

```
set theFile to choose file with prompt "Select a text file:"
set theFileReference to open for access theFile
set theFileContents to read theFileReference
close access theFileReference
```

As you can see from the example code above, the open for access command returns a reference to the opened file. That reference can then be used to refer to the opened file, using the read and close access commands. It is important to always use the close access command when you are done working with a file. Otherwise, the file will remain opened, and may not be opened for access again until it is closed. This can potentially produce error messages in subsequent runs of the script.

When reading from a file, the read command offers some optional parameters. For best results with these parameters, you should use the open for access and close access commands, along with the read command. The from and to parameters will allow you to read a small portion of the file's contents. For example, the following example code will read a file up until the 10th character:

```
set theFile to choose file with prompt "Select a text file:"
set theFileReference to open for access theFile
set theFileContents to read theFileReference to 10
close access theFileReference
```

The following example code will read a file between two specified characters, in this case, the text between character 10 and character 20:

```
set theFile to choose file with prompt "Select a text file:"
set theFileReference to open for access theFile
set theFileContents to read theFileReference from 10 to 20
close access theFileReference
```

The until parameter will allow you to read a file until a specific character is detected. For example, the code below will read a file until a return character is detected.

```
set theFile to choose file with prompt "Select a text file:"
set theFileReference to open for access theFile
set theFileContents to read theFileReference until return
close access theFileReference
```

The using delimiter and using delimiters parameters will allow you to read a file using one or more specified delimiters. The result will be a list of strings, broken apart by the specified delimiter(s). This may be useful when reading a tab-delimited file directly, as it would allow you to break apart the file as it is read by the script. For example:

```
set theFile to choose file with prompt "Select a text file:"
set theFileReference to open for access theFile
set theFileContents to read theFileReference using delimiter
tab
close access theFileReference
```

Some of optional parameters shown above possess additional functionalities that were not covered in this article. In addition, the read command also includes some other optional parameters, which may be useful in other situations. I encourage you to spend some additional time becoming familiar with all of the optional parameters of the read command.

Writing to a File

To write data to a file, you use the write command, also found in the *File Read/Write* suite. When using the write command, it is always necessary to use the open for access command prior to writing to the file. You cannot write to a file unless it has been opened first. In addition, when opening a file for writing, you must also specify the with write permission optional parameter for the open for access command. Otherwise, the file will be opened, but you will not be able to write to it.

The following example code will prompt the user to enter some text, and then write that text to a file on the desktop.

```
set theText to text returned of (display dialog "Please
enter some text:" default answer "")
set theFilePath to (path to desktop as string) & "test.txt"
as string
set theFileReference to open for access theFilePath with
write permission
write theText to theFileReference
close access theFileReference
```

Like the read command, the write command also has some optional parameters, including the starting at parameter. This parameter will allow you to specify at what point in the file to begin writing. By default, the write command will start writing at the beginning of a file. To start writing at the end of a file, you may use the term eof, for end of file. You may also specify a numeric value for the starting at parameter, specifying the number of the character at which the script should begin writing.

The following example code will append a specified string to the end of a file:

```
set theText to text returned of (display dialog "Please enter
some text:" default answer "")
set theFilePath to (path to desktop as string) & "test.txt"
as string
set theFileReference to open for access theFilePath with
write permission
write theText to theFileReference starting at eof
close access theFileReference
```

Optionally, you may want to use the set eof command, also found in the *File Read/Write* suite, in order to change the location of the end of the file. For example, the following code will set the end of the file to 0, wiping all existing content, prior to writing the new text.

```
set theText to text returned of (display dialog "Please enter
some text:" default answer "")
set theFilePath to (path to desktop as string) & "test.txt"
as string
set theFileReference to open for access theFilePath with
write permission
set eof of theFileReference to 0
write theText to theFileReference starting at eof
close access theFileReference
```

In Closing

Now that we have explored some of the ways that you can manipulate text content, you can begin to experiment with these methods, and combine them together in order to perform more robust types of processing. For example, try creating a handler that will write specified content to a text file. Then, call that handler throughout a script to maintain a running activity log. You may find such a log to be useful in monitoring the script's activity, as well as for troubleshooting purposes.

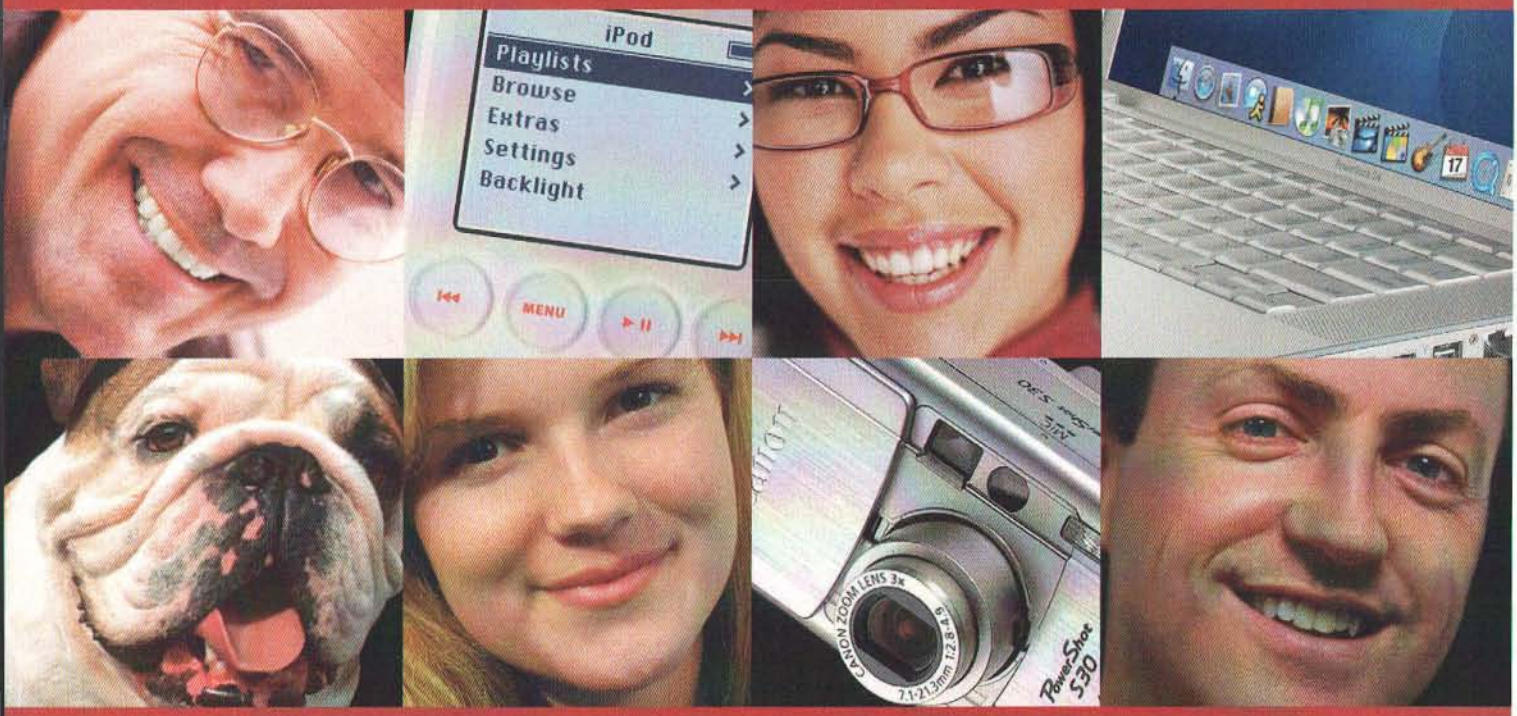
For continued learning about working with text, be sure to review the *AppleScript Language Guide*, mentioned earlier. You will also find detailed documentation and additional examples in most AppleScript books, such as *Danny Goodman's AppleScript Handbook*, available from *SpiderWorks, LLC* at http://www.spiderworks.com.

Until next time, keep scripting!

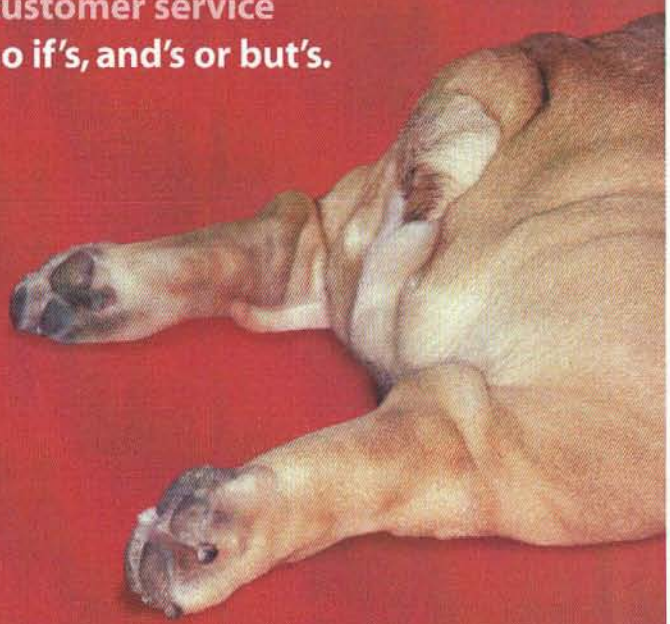
About The Author

Benjamin Waldie is president of Automated Workflows, LLC, a firm specializing in AppleScript and workflow automation consulting. In addition to his role as a consultant, Benjamin is an evangelist of AppleScript, and can frequently be seen presenting at Macintosh User Groups, Seybold Seminars, and MacWorld. For additional information about Benjamin, please visit http://www.automatedworkflows.com, or email Benjamin at applescriptguru@mac.com.

At Small Dog Electronics, happy customers are our highest priority.



When you shop at Small Dog Electronics, you get more than just great selection and low prices; you also get personalized service from genuine Apple Professionals who take customer service very seriously. And that's a promise... **no if's, and's or but's.**



Small Dog Electronics

www.smalldog.com

1-800-511-MACS

Apple Specialist

A socially responsible business since 1996

DNS And E-mail

Did You Know They're Related?

It is my charge to bring you all things Unix. To help you understand what's going on beneath the GUI. Sometimes, this involves troubleshooting. While DNS and e-mail no longer play exclusively in the Unix realm, they certainly spent their childhood there and have some Unixy-like scars to show for it. Not only did they grow up in the same era, they just happen to be first cousins. This particular column aims to be a tiny troubleshooting guide.

Family Tree

"DNS and *e-mail!*" I hear you ask. "They are such different systems, and do completely different things!" Ah! It's the differences allow them to complement each other, which is why they are such perfect mates. E-mail relies on DNS. In fact, most systems nowadays have some reliance on DNS. E-mail just won't get delivered without DNS. Kerberos not functioning? Check your DNS. Most other systems would just be cumbersome, as you'd have to specify other systems, not as a name, but as an IP address.

E-mail servers rely on DNS in several ways. Your outbound e-mail server must have access to DNS to lookup mail routing information. To receive e-mail, your Internet facing DNS servers must be set up correctly to let others know how to reach you. DNS mis-configuration is commonly a source of e-mail trouble.

DNS

Life without DNS cumbersome? Well, yeah – this is why DNS exists. Computers like numbers, people like names. The Domain Name System exists primarily to convert names, like www.apple.com, into an IP address (number), such as 17.112.152.32. It also helps us convert from number back to name. DNS has been pushed to do even more than these simple translations.

Bottom's Up!

I'm going to introduce these systems youngest first. Before DNS existed, the name to address translation took place by looking up entries in a flat-file called 'hosts'. This file still exists, and can still be used. There was a time when OS X ignored the hosts file, but it is now recognized by default. You can find it in /etc, where all good Unix boxes store the host file. When the Internet was small enough, current host files were copied between machines – actually, one master machine, via ftp.

Naturally, as the number of machines on the Internet grew, this system just didn't scale. In 1984, Paul Mockapetris released RFC 882 and 883 – the Domain Name System. Naturally, these RFCs were supplanted and augmented, but those two were the birth of DNS.

DNS provides a hierarchical, distributed database. Name servers are the programs responsible for transferring the database to clients (as appropriate) and to other servers. OS X ships with BIND (Berkeley Internet Name Daemon) as its name server. There are other servers, such as DJB DNS, MaraDNS, and even stand-alone

DNS hardware such as Bluecat DNS. Tiger server, specifically, runs version 9.2.2 of BIND. While there are pros and cons to BIND, it is certainly the most popular name server on the Internet.

Enough history, on to the details! Please note that it is outside the scope of this article to dig into every facet of DNS, and I'll just be hitting on the basics (remember, we still have to get to e-mail!). If you administer a name server, need to know more about BIND or even walk by a machine running a name server, you owe it to yourself to pick up "DNS and BIND" by Albitz and Liu, published by O'Reilly. Best. DNS. Book. Ever. I still have my worn 2nd edition. The only reason I haven't picked it up in a bit is because I committed it to memory long ago.

The Details, Please!

Most techies (that's us) understand DNS structure. Generic top-level domains, such as 'com', 'mil', 'info' and 'org' hang just off of the root (".") . Delegated zones live beneath the gTLDs, and subdomains and hosts follow. When queried about a zone that it operates (or, is ***authorative*** for), the name server will consult its database. BIND uses a simple flat-file scheme for a database, that is read at name-server start up. Each entry in the database is referred to as a ***record***. There are many different kinds of records, and I won't be covering them all, just the ones germane to this discussion.

The main record to understand immediately is the A, or "'address"' record. This provides the core functionality: it maps a name to an address. An excerpt from Apple's DNS may look something like this (the "'IN"' stands for "'INternet class"' – somewhat of a relic, but still a requirement):

```
www.apple.com.          IN A 17.112.152.32
mail-in3.apple.com.     IN A 17.254.13.8
nserver.apple.com.      IN A 17.254.0.50
```

So, when you request www.apple.com in your browser, your system asks DNS to ***resolve*** the name, and in turn is told '17.112.152.32'. (For those of you who know this cold, yes, this is a simplified version of how this all works). You can use DNS to help transition to a new server by changing the A record. Let's say you run web services at www.example.com. The A record for www points to 10.0.0.10. One day, you buy the biggest, shiniest new server. You could set it up as 10.0.0.11, configure it, and once ready, alter DNS to map www.example.com to 10.0.0.11. People don't have to learn a new name, buit they will get routed to the new server.

Didn't You Say Something About Mail?

Move all of that DNS information over in your brain for a moment; I'll tell you when to slide it back into the center again. E-mail has long been one of my favorite systems. I'm not sure why. It absolutely disturbs my inner core when an e-mail system is not working. I started off with sendmail (before the m4 macros were the way to write your cf files), and shifted to Postfix when it was still Secure Mailer. What is seemingly a very simple thing – moving a text file from point A to point B – actually has a fair amount of complexity.

If you've only ever configured the mail system via the GUI, or come from an Exchange, Kerio or Communigate background, there's a simple issue that you may not know about. The subsystem that is responsible for sending and receiving mail is ***completely separate*** from the subsystem that allows you access to your mail spool. When someone from the outside sends mail that is destined for your mail server, Postfix is responsible for receiving that mail via the SMTP protocol, figuring out what to do with it, and either delivering it personally, or handing it off to an intermediary. When you set up a mail client (or "MUA" – Mmail Uuser Aagent) to retrieve mail via the POP protocol ("boooooo!") or the IMAP protocol ("Yeah!"), Cyrus is handling those requests. When you send an e-mail via your MUA, be it Entourage, Mail.app, Eudora, whatever – once again, Postfix is responsible for talking to your mail client, queuing and sending the mail to the remote system. So how does this all tie together?

Once again, imagine you run all services for example.com. When mail destined for mail.example.com is presented to your OS X Server, Postfix, listening on port 25 for activity, springs into action. Postfix itself has several subsystems that deal with various stages of accepting, queuing and delivery. However, one of the first things Postfix must do, is decide whether or not this piece of mail belongs on this server or not! Is it really for 'example.com'? If so, is there a user by the name requested? Important questions, right? If the mail truly belongs here, it has to get added to the correct user's mail spool. In OS X's case, Postfix hands the mail off to Cyrus via LMTP.

Most people have heard of SMTP, the simple mail transfer protocol. This protocol is responsible for delivering mail between two systems over a network – typically over a WAN or the Internet. Somewhat lesser known is LMTP, the local mail transfer protocol. OS X uses LMTP exactly as designed: to allow two mail systems residing on the same machine to exchange mail (actually, these two ***can*** also be two separate machines on the same network). LMTP uses the same semantics as SMTP, but avoids queuing. LMTP ***must*** process each message as it isthey are received. It is also stipulated that LMTP must ***not*** run on port 25. In our case, LMTP exists as a socket in /var/imap/socket.

Once the message is handed off via LMTP, Cyrus is responsible for dropping the message in the proper user's mail store (specifically, Ccyrus' 'deliver' agent does this, although nowadays, 'deliver' is deprecated, and is just a wrapper for LMTP delivery). When a person commands their e-mail program to "'Get New Mail'" Cryrus is responsible for answering the request by accessing the mail store, and answering via the POP or IMAP protocols.

Uh-oh

You're a freelancer or in house IT person that's responsible for a mail system, and you get the call: "I'm not receiving any mail!" What to do? First, find out of it's just the person who called, or if the problem affects everyone. If you have an account on the system in question, send yourself a test message from your GMail account (unless, of course, someone from Google is reading this and you're trying to troubleshoot GMail. In that case, use Hotmail). If you receive it, great. It's likely limited to that one person that called.

If you didn't receive it, time to start looking at the logs. Let's stop right here for a second while I admit something. I love logs. I'm a bit obsessed with them. A future column will touch a bit more on logs, but let's just remind ourselves how important they are. Logs are the heart of your system. I typically keep one machine nearby that does nothing but watch logs for me. It's good to glance at so you get used to the rhythm. That way, when that rhythm gets broken, you can recognize it instantly. That said, you want to follow /var/log/mail.log right now. I recommend using 'tail -f' in terminal for this. Send that test message again. Is there an entry in the log for this?

OK, now's the time to dust off that DNS information. If there's no entry in the mail log, it's likely that the mail server is just fine, rather, it's DNS that is having an issue. As mentioned, there are several kinds of DNS records, the A record just being one. The MX record defines a 'mail exchanger' for the domain or subdomain. If that's not configured properly, mail isn't going to get delivered.

When I type up and send an e-mail, my MUA hands off the message to my e-mail server (or, Mail Delivery Transfer Agent – MTDA). The mail server then queues the message and figures out how to deliver said message. If my message is destined for steven@radiotope.com, the server must first find out where it is to send this mail. DNS has the answer. This is a three-part conversation that would go something like this:

example server: "Excuse me, DNS for radiotope.com, can you please give me your mail exchanger?"

radiotope DNS: "Sure, it's www.radiotope.com."

example server: "Oh. I've never dealt with that server before, what's its IP address?"

radiotope DNS: "No problem, that's 69.55.224.105"

example: "Thanks!"

example: "Pardon me, www.radiotope.com, I've got some mail for you!"

radiotope mail server: "I'll take that, thank you!"

Notice how much of that conversation happens with the DNS server. Fortunately, the mail server will ***cache*** the results for radiotope.com, and won't have to ask until the cache TTL (time to live runs out).

So, if DNS isn't correct, the mail will never hit the proper mail server, and you'll never see it in the log file.

I've had an incredible increase in the amount of mail server troubleshooting that I'm doing for people. Apple certainly hasn't had a perfect record here. Up until 10.3.4 in the Panther series, people were bitten with the 'log rolling bug'. That did get fixed. However, we're all still living with the 'reconstruct bug'. When you check the mail.log file, you may see entries like this:

```
Jul  5 10:33:38 mailserver deliver[379]:
connect(/var/imap/socket/lmtp) failed: Connection refused
```

Or, this:

```
Jul  3 04:49:27 mail postfix/pipe[8805]: 6E8AF9A9F2:
to=<jimbob@example.com>, relay=cyrus, delay=135898,
status=deferred (temporary failure. Command output: couldn't
connect to lmtpd: Connection refused_ 421 4.3.0 deliver:
couldn't connect to lmtpd_ )
```

On a busy mail server, you'll see ***a lot*** of these. They're both commonly associated with a corrupt Cyrus database. Time to stop mail, and for a little explanation. Stop the mail services:

```
serveradmin stop mail
```

Although the issue here is with Cyrus, this command will stop both Postfix and Cyrus. You don't want things trying to deliver to a corrupt mail store. As mentioned earlier, Postfix hands the message to Cyrus via LMTP via a socket file. It was also mentioned that LMTP doesn't perform queuing – it is required to process each message as it's received. In this case, it can't. Either Cyrus isn't running at all, or it has corruption in its database. To see if Cyrus is running, look for the evidence of 'master' in your process list:

```
# ps ax | grep [m]aster
  206 ??  Ss     0:00.01 nfsd-master
  227 ??  Ss     0:57.27 master
29133 ??  Ss     0:00.05 master
```

Whaaaa? There's two! For better or worse, both Cyrus ***and*** Postfix have their master processes named "master". You can see which is which with lsof:

```
lsof | grep master
```

You should see a group of master processes owned by root, followed by a group owned by cyrus (cyrus-imap on Tiger). If Cyrus is genuinely not running, watch your system log (/var/log/system.log) and hand crank it: "/usr/bin/cyrus/bin/master &". Naturally, you can also stop and start all mail services while watching the log.

Cyrus

Unlike many other IMAP and POP servers, Cyrus takes a different approach to handling mail. Traditional IMAP

servers, such as UW-IMAP, really don't get involved in the delivery process at all. You tell them where the mail spool is, and they relay that to the user. In some cases, the IMAP server will be responsible for taking mail out of the system spool and moving it to a spool in the user's home. Not so with Cyrus. Once Postfix hands off the message to Cyrus, the 'deliver' agent is responsible for getting the mail into the correct mail queue.

Here's where Cyrus is completely different. Deliver will drop the message into the correct user's mail store, sitting at /var/spool/imap/user/(username) – this is the default location for OS X, and you can move it to another partition, so be aware if you have done so. "deliver" is Cyrus' Mail Delivery Agent (MDA). That's all standard, but Cyrus goes above and beyond this, keeping a database of mailboxes, ACLs (not the Apple filesystem ACLs: – Cyrus sports its own ACL list that can let others have access to your mailboxes. As far as I know, there is no Apple-official way to change the Cyrus ACLs), seen messages, quotas, etc. The databases that keep track of this are in BDB format, and live in /var/imap.

Like any database, there's always a chance that it will get corrupted or somehow out-of-sync. Sometimes, I've seen this happen with a system crash. OK, that makes sense. The disks went down dirty, and files didn't get closed. Oddly, I've also seen Cyrus go batty with no great explanation (however, I do sometimes show up on the scene to clean up, and let's face it, people aren't always 100% accurate about the events that lead up to the whole mail system going down). But if you're seeing errors in the logs like the two above, there's a good chance you have some Cyrus database problems.

Cyrus database problems can come in several flavors, so let's hit the highlights:

Permissions errors

Our good old friend 'permissions' is back! /var/imap should be owned by cyrus:mail, with perms of 755, all the ways down. Same goes for /var/spool/imap, with one exception: /var/spool/imap/user should be a little more restrictive: 700 for it and its contents.

If Cyrus can't read and write into those folders, it's going to have problems delivering and retrieving mail. So double-check those folders!

Damaged Socket File

You should really never see this condition. However, this is a variation on the first condition: the socket could have the wrong permissions assigned (although, that should never change unless you touch them!). Bonus: they auto create on Cyrus startup. Just nuke 'em from /var/imap/socket with mail services stopped. They'll get created properly when services are restarted. For the record, a Tiger server at minimum will look like this:

```
# ls -l /var/imap/socket
total 0
-rw- - - - 1 cyrusima mail 0 Jul 2 00:12 imap-1.lock
srwxrwxrwx 1 root     mail 0 Jul 2 22:50 lmtp
```

Database corruption

This seems to be the big one. There are several things that can go a bit batty here. I can't stress the importance of backups enough. The crummy thing is that to back up Cyrus properly, you need to shut down mail, and there's no great alternative on OS X. Hot backups here can lead to inconsistent states. Backup, backup, backup, and test, test, test! I mention this simply because restoring the database from the mail spool is really imperfect. There are certain things that can ***not*** be inferred from the mail store, so those things get set back to a default state.

Here's how this works: as Cyrus drops mail into the mail spool (/var/spool/imap) appropriately, it updates its databases (/var/imap). They're relatively independent – the mail spool can live without the database, however, that's where Cyrus stores all of the metadata (remember that stuff?), so it needs to be protected. Cyrus keeps watch over this by check-pointing the database at regular intervals. The tool that rebuilds the database is called 'reconstruct'. I'll spare you the long story and cut right to the chase: Apple's latest version of reconstruct is a bit FUBARed. The bad version snuck in under a 'security update'. There's an open source replacement that corrects the mistake that Apple's version makes. Get it here: http://www.sussex.ac.uk/Users/jamesg/reconstruct.zip. The (long) explanation for this takes place in this thread on Apple's support boards:

http://discussions.info.apple.com/webx?7@1022.JKM0abd8V6f.2@.68aec789/9

Is there any harm in running Apple's version? Well, yes. Apple's version will reset the internal time-stamp of messages to '0' (zero). This won't affect every mail client, but guess which two it will? Apple's Mail.app and Entourage – the two most popular mail client on OS X. How do you know if you've been hit with this? If, after a reconstruct, your mail clients are choking while trying to check for new mail, but webmail (using the built-in squirrelmail) is fine, sounds like this is the issue. Get the replacement from the URL above, choose the version appropriate for your platform (yes, this currently still exists in 10.4.1). Backup your original (mv /usr/bin/cyrus/bin/reconstruct /usr/bin/cyrus/bin/reconstruct.apple) and drop the replacement into the same directory (/usr/bin/cyrus/bin/reconstruct). Set the wheels in motion like this:

1. Start a pot of water boiling. Now is no time to panic, and tea will do you some good. Preferably some green or camomile.
2. Stop all mail services. If you didn't do so above, do so now: serveradmin stop mail
3. backup /var/spool/imap and /var/imap...just in case
4. Run reconstruct. You must do this as cyrus:

```
sudo -u cyrus /usr/bin/cyrus/bin/reconstruct -i
```

The '-i' flag here will rebuild sub folders for each user as well. You should see messages scrolling by as reconstruct works its way through each user. Depending on the size of the mail store, this could take some time. Now's the time to pour that cup of tea.

Once that is complete, try firing up the mail system again: `serveradmin start mail`. On rare occasions, this won't start Postfix. Sip that tea, and simply type: postfix start. No problem. At this point, mail should be flowing again. Queued mail, and new mail coming in should be getting delivered, and people should be able to access their mail. This alone will make them happy enough to ignore this issue: ***ALL*** of their mail will now be marked unread. Flags will be lost. But! Their mail will be back! Make the rounds, then finish off that tea. Maybe even get a cookie. What the heck.

Redundant Redundancy

A brilliant concept with DNS and mail is that you can specify ***multiple*** MX records. You can (and should) assign them a ***priority***. "0" is the highest priority, with everything else being lower. So, when DNS specifies:

```
example.com.  IN MX 10 ren.example.com.
example.com.  IN MX 20 stimpy.example.com.
```

"10" is the ***higher*** priority server. That's where mail ***should*** go. However, if ren.example.com is down, the mail will get delivered to stimpy.example.com. For best effect, ren and stimpy should be on separate networks. In different buildings. In different states, if possible. If you run mail services in-house, see if your ISP will perform backup MX duties for you. Most will. Like backing up data nightly, having a backup mail server (or two...or three) is more than worth it. Don't walk this tightrope with no net: have a backup mail server with properly configured MX records.

In the previous section, we had to take our mail services out of commission while we repaired the Cyrus databases. If someone was trying to send us mail at that moment – something that has a very high probability – without a backup mail server, it's going to bounce back to them. We'll have dropped off the face of the Internet! However, with the backup server in place, the mail will get routed there, and be delivered to our main mail server once it's back on-line. Brilliant, just brilliant!

Other Troubleshooting

Naturally, for a mail server to receive mail, it must be accessible. I recommend that anyone who does network troubleshooting have an externally accessible machine –

preferably through ssh, although ARD and Timbuktu will let you get a shell also. This will be used for the purpose of getting an outside view of the network you're currently working on.

To test SMTP, access your outside machine, get a shell, and type:

```
telnet mail.example.com 25
```

Of course, you need to substitute the Fully Qualified Domain Name (FQDN) of *your* mail server. You should receive a reply like this:

```
Trying 192.168.0.18...
Connected to mail.example.com.
Escape character is '^]'.
220 mail.example.com ESMTP Postfix
```

Type 'quit' and press return to get back to your prompt. If you do not get this greeting, either Postfix is not running, or something is blocking access. If your external test machine is on a residential network, it's likely that port 25 is blocked. Make sure you don't let that throw you off. You can have Postfix listen to both 25 and an alternate port by adding this line into /etc/postfix/master.cf:

```
8025    inet  n       -       n       -       smtpd
```

Where 8025 is the new port that Postfix will listen to. Save the file, and type:

```
postfix reload
```

to restart Postfix and allow it to incorporate this change.

You can test IMAP in a similar way. Again, using telnet, connect to port 143:

```
# telnet mail.example.com 143
Trying 192.168.0.18...
Connected to mail.example.com.
Escape character is '^]'.
* OK mail.example.com Cyrus IMAP4 v2.2.12-OS X 10.3
server ready
```

Just like the SMTP test, if you do not see this greeting, either Cyrus is not serving up IMAP for some reason, or the port is being blocked from you. While I haven't found an ISP that currently blocks IMAP, perhaps there is an errant firewall rule preventing access. When you're done here, type "01 logout" and press return.

Serendipity

Like this column, e-mail and DNS are completely intertwined. Both have nuances and pitfalls that require a bit of knowledge. I've been helping people out more and more frequently with mail issues. The sad news is that OS X's built in mail system is just about there, but not quite. The problems lie more on the Cyrus side – specifically due to the way they're implemented on OS X. You'll find plenty of stable Cyrus servers running on other platforms. Postfix, thankfully, is just a rock. When troubleshooting mail, remember: mail is really a bag of tricks, and not one single coherent system. There are, of course, other factors that will impact mail delivery, such as network design, choice of mail host, content filtering schemes, intermediary mail routers, firewall and/or NAT rules, and more. Many facets of your seemingly distinct pieces of network equipment are inherently intertwined.

This isn't always the easiest of stuff. Questions? Feel free to send 'em my way at emarczak@mactech.com.

About The Author

Ed Marczak, owns and operates Radiotope, a technology consulting company that implements mail servers and mail automation. When not typing furiously, he spends time with his wife and two daughters. Get your mail on at http://www.radiotope.com

Game Controllers

Some Different Ways To Kill The Creatures

The keyboard and mouse. Your basic gear for getting the bad guys, the bosses, and cartoon thingies that burst into clouds of flowers and hearts in their death throes. It works. Quite well, actually. But, every now and again, it's just not enough. Sometimes, you just want to feel like you're eight, and spending your allowance at the local arcade, but do it on your 30" wide screen Cinema Display. Or you'd like to mop up that level full of demons using your game console controller, if only you could plug it in. Well, wish no more.

X-Arcade

So, like I mentioned above, every once in a while, I long for the simple days of dropping quarters into a really big boxes, and blasting long lines of pixilated aliens for about 2.5 minutes before getting myself blasted three times in quick succession. Okay, so maybe not all that often, but it does happen. I got one of those little gadgets for Christmas that plugs into the TV, and it was great. But, it didn't control the way I remember. Well, take a look at this. (See Picture.)

Pictured above is the X-Arcade dual control. There is also a single control version. Oh, my. It's like they went to the scrap heap, yanked the control boards off every old Asteroids machine they could find, polished them, and shipped them off to their customers. In actuality, that's not far off. The X-Arcade folks have built what is, really, the console from any arcade video game you played as a kid. The joy stick's feel is genuine. The buttons are dead on. The box is built out of the same materials as the real thing. It's brilliant. It looks and feels exactly like the real thing.

It is designed to play the old arcade games. MacMAME is all ready to play with it, which is quite nice. Load it up, get some ROM's, and you're in business. X-Arcade's web site has links to some legal ways to get games to play with it. However, the one thing that drove me nuts was that the buttons were all the same. I ended up taping labels next to the buttons in order to know what I was doing. Color coding them, or numbering or lettering them would have been a very nice idea.

www.x-arcade.com/mac/

Nyko AirFlo EX

"Dad! Come get this boss for me! I can't! Pleeeeeaaaaasssssseeee!?" Yeah, I

know, you've heard it, too. You then trudge up the stair, only to be handed this slimy, wet thing with bits of who knows what snack dripping off it because your spawn is midway through a weekend long game finishing marathon. Ick doesn't quite cover it. Well, thank goodness for Nyko. They've got out what is essentially a standard PlayStation style controller, USB, that's got one thing most other controllers don't have. Air conditioning. Think on it. The AirFlo EX has a built in fan that runs air through the handles, and out the vented holes of the thing. Outstanding. Grip it for hours on end, and you won't have a bit of sweat on the controller. The feel is very like that of a PlayStation controller, except for a few things. One, the grip is knobbed, making it hold better. Also, in order to fit the fan in, and leave enough room to move air through it, it's rather bigger that the standard sized controller. It actually a bit more comfortable for bigger hands, but small kid's hands (referenced above) may find it a bit much over time. My only other gripe is the same one I had for the X-Arcade. The lack of labeling on the buttons. I know they can't put the proper PS2 colors and shapes on the button (copyright and all that), but does make it a little more difficult.

The drivers for it are rather nice. It's clean, works as you'd expect, and runs under Tiger. 'Nuff said.

www.nyko.com

By Michael R. Harvey

Final Cut Express HD

I was sitting in the audience at Macworld when Steve Jobs announced iMovieHD. I've gotta tell ya, at the time you could have heard a pin drop. I turned to my buddy and whispered, "***High-Def? I don't even work with high-def and I'm a professional video guy! What's my mom going to do with HD?!?***" It was one of those moments when Apple had jumped so far ahead of us all that we couldn't make out the taillights any more.

Since that day, HD has been picking up so much momentum that you can't pick up a newspaper or watch TV without seeing something that's selling HD-ready features. Granted, the cameras are still a bit pricy, but the prices are coming down fast. For example, Sony's offerings have gone from $20,000 to $3,500 with their HDR-FX1, and as I'm writing this, they just announced the HDR-HC1, a $2,000 addition to their HandyCam lineup that makes things even more interesting.

Now let's say that you drop the cash on one of those cameras and iMovie isn't going to keep up with your creative needs, but you're not really needing the full capabilities of Final Cut Pro 5. That's where Final Cut Express HD (FCE) fills the void. Final Cut Express HD is version 3 of Apple's mid-level editing package. This $299 software package will do about 80-90% of what people do with Final Cut Pro for 1/3rd of the cost. As for the rest of the FCP's features, most users won't really ever think to look for them unless they're moving into the professional arena.

For starters, don't let the HD freak you out. Personally, I still don't own an HD camera of my own (although I have a couple I can borrow in a pinch) and odds are you don't either. Think of FCE's High Definition ability like that optical jack on today's stereos. If you use it, you get this incredible surround sound quality, but if you just use the same old analog white and red (left/right) cables that you've used for decades, you still get good sound, just not *"WOAH NELLY"* surround sound. FCE works just fine with the MiniDV footage we've all come to know and love. When you find yourself with an HD camera, you'll get a much better picture than you're used to, but you'll import it, edit it, and display it pretty much the same way you have with standard definition television. It'll just take a little longer for some of the filters to render.

The typical FCE user will be buying the software because iMovie leaves them unfulfilled. With that in mind, Apple's made it seamless to open up your iMovie projects in FCE. All you have to do is hit ?-O to open a project, then select your iMovie file. You'll find yourself looking at all the work you've already done in iMovie, but now you can take it to the next level with FCE. Any video transitions you used in iMovie will also show up in the timeline, but they will come in as separate, rendered video clips instead of transitions. Many iMovie transitions don't have FCE counterparts, so if your artistic vision **requires** the use of ***disintegrate,*** you'll have to work in both programs.

Importing footage is a cakewalk, as long as you're not mixing formats. If your project is an HDV format project, FCE will ignore your NTSC-DV format deck or camera, and tell you that it can't find the HDV deck, which can be a bit confusing. Essentially, FCE is built to work with one format and then stick to it. You can mix HD and SD footage in the same project, but you'll have to import the footage using 2 different projects first, one for SD and one for HD.

Most of the editorial features are virtually identical to FCP, so if you're familiar with the Pro version, you'll take to Express like a duck to water. The iMovie to FCE converts will be gaining an incredible amount of control over their video from what they're used to. They'll also be getting an equally incredible number of new buttons and menus and tabs. Be prepared for a learning curve, but trust me, once you learn how to use all the features of FCE, you'll be armed with abilities we only dreamed of back in the old edit bays of the 20th century, well, prior to 1990 or so at any rate. Not only that, but if you master FCE, then you'll have the technical skills to be a professional video editor. I'm not saying you'll be any good, but you'll ***technically*** know how to do the work. (I know how to work a paintbrush, but I ain't no VanGough.)

The key feature (pun intended) that's missing from FCE is key frames. When you see a logo drifting across the background, that's using key frames. You set a start location key frame and an end location key frame, and FCP will move the item from A to B. You can't do that

with FCE. It's more than a little disconcerting for us FCP users at first because we're so used to seeing the key frame areas in many of the tabs. With FCE, you apply a setting, and live with it for the whole clip. Of course, you can overcome this somewhat by using dissolves between 2 versions of the same clip; for instance, dissolving from a desaturated clip to a normal clip will let you go from black and white to color over time, but you can't drift objects, or create a dramatic rack focus, or do the thousands of other obscure special effects that key frame tinkering can produce. If you're getting that fancy, pony up for FCP.

Probably the single biggest ability you get with FCE that you can't do with iMovie is the ability to work in layers. FCE will let you stack several layers of video on top of each other to make a complex video image. Have a look at the graphic in the SD vs. HD sidebar, and you'll see something I completely created in FCE. The pictures, the text, the red border, and the drop shadows are all from FCE. It makes a nice still, but it's more impressive when the video is playing.

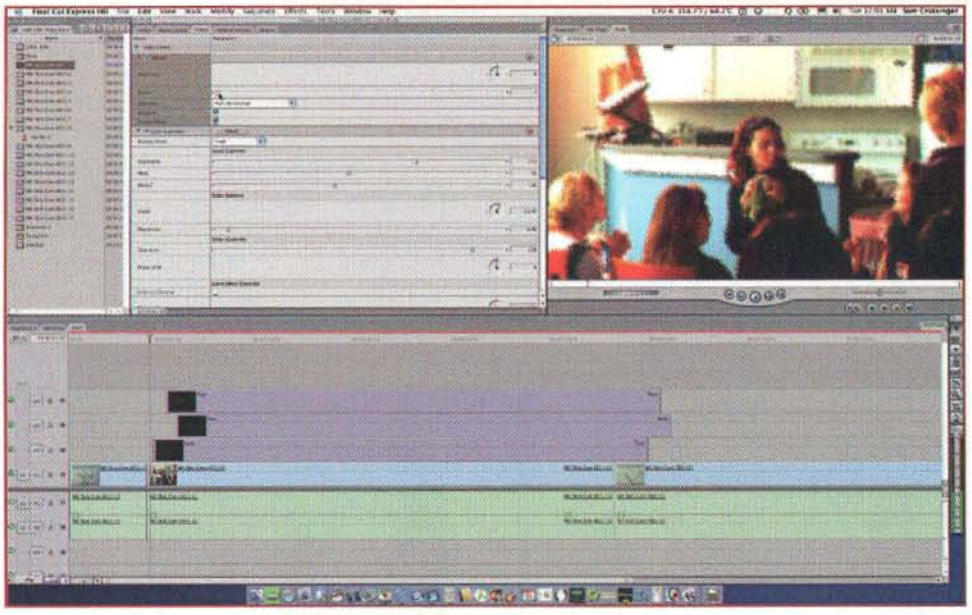


Figure 1: The FCE HD interface

FCE will let you work with pretty much anything that QuickTime can understand, so you can add TIFFs or JPGs to your movie, or you can use FCE to make a slide show movie. Unfortunately, you can't do that Ken Burns Effect stuff in FCE because of that pesky key frame thing. You ***can*** import a Ken Burns Effect iMovie into FCE, but the pictures will come in as rendered video clips instead of stills at that point, so you can't change their drifting properties any more.

Something that has FCE sitting on the fence is the use of time code. Time code is what video editors use to map out exactly what frames of video they want to use in a sequence. Each frame of video has it's own number based on Hours:Minutes:Seconds:Frames. FCE is a simplified version of FCP, so obviously it uses time code to keep track of everything, but it has nothing to do with the time code on your source tape. All DV camcorders record time code on the tape so you can find an exact frame of video by the numbers. FCE doesn't import that time code when it imports it's video. It edits by ***timer*** values instead. Every clip starts at 00:00:00:00 and counts up. It doesn't matter if it's the first clip on the tape or the last clip on the tape. They all start at zero. Because of this you can't produce Edit Decision Lists. You also can't import your footage at low resolution and then batch digitize it at high resolution later. If you're coming to FCE from iMovie, you won't really care. You can still get the job done just by looking at the picture, and picking what frame you want to start with just like you do in iMovie. If you're used to FCP, then you'll have to retrain your brain to get away from the numbers.

That last point bears repeating. Footage captured with FCE ***has no time code track at all.*** Learn from my mistake. You might think it would be fantastic to use the cheaper FCE for importing footage while you keep working on the more expensive FCP system, but **don't do it!** If you ever have to go back to redigitize the footage later, you'll realize that you can't. Professional projects need to stick with Final Cut **Pro**.

However, you can go the other way as far as I've been able to tell from my testing. FCE might leave you high and dry when it comes to importing footage, but it does a fine job of ***rendering*** timelines from FCP. Take this with a grain of salt, but it worked for me. If you have FCP and FCE both installed on your system, and you open an FCP

project with FCE, you can let FCE render the timeline in the background while you keep plodding away in FCP in the foreground on another project. You'll need a fast computer, but for some of us, it's worth $299 to be able to render in the background so we can keep working. It even renders files that can't be created with FCE because of key frames or whatnot. That's very cool.

A few of the other things that fall into the *pro* area which aren't in FCE: Audio mixer, 3-way color corrector, video meters (waveform, vectorscope, etc.), time code generator/reader, Media Manager.

A few really cool things that *are* in FCE that I wasn't expecting to see:

Chroma Keyer & Color Keyer - for knocking out green screen backgrounds.

Basic 3D - lets you position objects at any angle to the video plane, but without key framing, you can't have an object rotate in space over time.

Image Stabilizer - takes minor shaking out of footage.

(In fact, almost all of the filters and transitions found in FCP are in FCE.)

Sidebar:

SD & DV vs. HDTV & HDV

The television format we've all been watching since Ricky loved Lucy is now being called ***Standard Definition (or SD or SDTV)***. It only got that name after ***High Definition (or HD or HDTV)*** video came along. The difference mostly boils down to resolution. HD has 6x the picture resolution of SD, which means it eats up a lot more hard drive space. DV footage (3.6 MB/s) is compressed a little to make it more manageable. HDV footage (15 MB/s) on the other hand is pretty heavily compressed. Most of the time that compression will hold up pretty well, but if you find yourself at a New York fashion show with strobe lights popping all over the place and you move the camera around, you'll be able to see the codec choke on all the changing pixels as it puts ugly little boxes all over sweet, lovely Tiiu Kuik's face for a frame or two. (Are you catching on that I'm not being hypothetical about this scenario?) The moral of the story is to keep HDV away from images that border on berserk and it'll deliver damn fine pictures for the price. If you want perfect HD pictures, you're moving into the $60,000 camera ballpark. I think it's safe to say that most people will find a way to live with the limits of HDV.

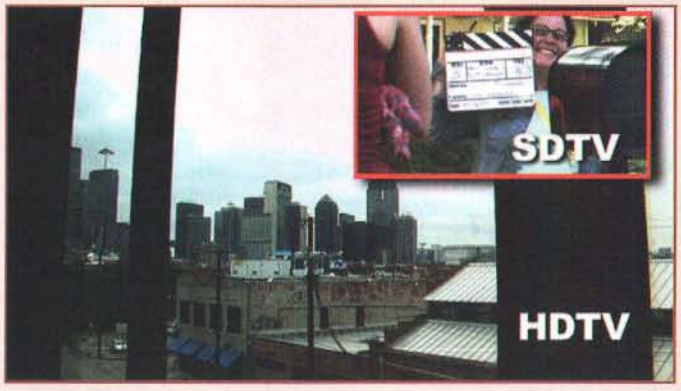


Figure 2: Widescreen (16:9) SD vs. HD image size difference

By Sam Crutsinger

About The Author

Sam Crutsinger is the Media Kingpin of TackyShirt, a company dedicated to producing training videos that people actually want to watch.

Old dog. New trick.

Growing up in Hawaii, Ron Stanford learned to surf at an age when most kids are still mastering the tricycle.

Making web sites came a bit later.

Ron's love of surfing took him to Nicaragua, where he fell in love with the uncluttered beaches and the locals' unassuming approach to tourism.

"I started NicaSurf.com in 1998, and the first year of producing it was sheer torture. I knew what I wanted the site to look like, but why was it so difficult to achieve?"

As an award-winning creative director and filmmaker, Ron was used to the control and finesse of desktop publishing and nonlinear editing tools. So he expected a lot from his Web design software.

"In 1999, a friend showed me Freeway. And a light went on—this is how Web design was supposed to work! Freeway lets me put my ideas on the page, exactly how I want them to look—it's total design freedom, no compromises. And I don't need to hire a big team just to get the job done."

"I was able to make a site with Freeway that expressed my love for surfing and Nicaragua. And now my client in San Juan del Sur has customers from all over the world."

Don't just surf the Web. Download a free 30-day trial of Freeway Pro or Freeway Express today, and make your own waves!

Winner MacUser Awards 2003

Freeway Pro
Web Site Design and Publishing

Macworld AWARDS 2004 WINNER

Freeway Express
Editor's Choice: Best Consumer Software

Recent Reviews:
MacDesign: 5 out of 5
MacFormat: 5 out of 5
MacDirectory: 4.5 out of 5

freeway

www.softpress.com/mt

Vendetta Online

Vendetta Online is the three dimensional, massively multiplayer online version of Escape Velocity. Really. It is at least as much fun as any of the three EV games are, and it is absolutely beautiful to look at. Not only that, it's a blast to play. When you first join the universe, you can run through several training missions to get you up to speed, and prepared for operating there. You don't have to, though. You can jump right in, and figure it out as you go along. There's something to be said for going out with nothing, and killing things without having a clue as to why. Figuring out docking with space stations was an adventure, too. But, you'll get it figured out soon enough.

And, figuring it out really isn't all that hard. The controls are set up along the same lines as other games of similar type. You're using the keyboard mostly, although you can use the mouse, or a joy stick. I found that going from the keyboard to the mouse was a bit more trouble than it was worth, except during combat.

The basic message to take away form this is that this game is all kinds if fun to play, incorporating elements of first person shooter, space shoot-em-up, role playing, and strategy gaming into this online fun house.

So, how do you come by it? Well, start here: www.vendetta-online.com. From here, you can find out more about the game, as well as the community. From here, you can also download the game itself, which weighs in at around 112 MB. You also have the option to buy a boxed copy, for around $10, and that includes a few extras, like a poster of the universe, and a booklet explaining the history.

Well, then, what about paying for playing? From the above mentioned web site, you can create an account, and get yourself a free trial of the game. Then, if it hooks you, which it just might, you're out $9.99 a month when you subscribe on a month by month basis. Sign up for longer terms, and the price drops, down to about $6.67 on a 24 month subscription. That is a great price, especially compared to other MMORPG.

Keep one thing in mind when deciding if it's worth it. The Vendetta universe is not static. The guys at Guild Software are continuously working on the game, expanding the place with more adventures to take on, and places to visit. That strikes me as extra cool.

Give it a go. You've got nothing to lose, and I bet you'll find it's great fun. And, if you're into them, this MMORPG is just about the best deal out there. It doesn't hurt that it's quite a good game, to boot.

By Michael R. Harvey

Point-and-Click Astronomy With A Mac

Use Starry Night, A Mac, And A Homemade Cable To See The Universe

Starry Night Pro is a fantastic piece of software for the astronomically curious, as well as the advanced amateur. It seems capable of doing practically anything, and the further you climb up the learning curve, the more really cool stuff it can do. The scope (pun?) of this software is so enormous that writing a proper comprehensive review would be a daunting task, so instead, this article will concentrate on one of Starry Night's functions that many observers can use and appreciate: Telescope control.

Equipment

For a number of years, telescope manufacturers, such as Meade and Celestron, have sold telescopes with computers and drive motors that can be controlled from either a handheld unit included with the telescope, or via external computer. The included handheld units provide adequate control ability, including catalogs of thousands of stars, deep sky objects, all eight other planets, and the moon. Computer control, however, enables a user to view a map of the sky from a custom location, and point-and-click to select targets. Is M57, the Ring Nebula, on tonight's observing schedule? Point, click, and have a sip of coffee while your telescope slews to the proper location. Use the computer to zoom in tight on the map to look for other faint fuzzies, and then slew again to a new extra–galactic treat. It's simple, it's geeky, and it's just plain cool.

Starry Night has the built in ability to talk to several kinds of telescopes. The telescope available for this review was a Meade LX200 12–inch Schmidt-Cassegrain, acquired new in 1997. Newer or older telescopes may have different capabilities or requirements, so interpret the content of this article appropriately.

The perfect computer to act as the controller in this scenario is, of course, a portable Mac! Any of Apple's thin, light, power–conscious notebooks with a G4 processor should provide enough horsepower to handle the myriad of calculations that Starry Night must perform every second to pinpoint celestial objects, and the light-up keyboard of a recent-model PowerBook is a blessing for those typing under the pitch black skies that make astronomy worthwhile. However, there's one catch: The communications port on the telescope doesn't match any of the communications ports on a Mac, or most other modern laptops for that matter, so an inexpensive cable has to built in order to get these two devices talking.

Cable Construction

At the telescope's end of things, a type of connector called an RJ12 is needed. An RJ12 is a connector nearly identical to the connector at the end of a phone cable, except instead of having four conductors, it has six. A pair of RJ12 connectors crimped onto the ends of a common phone cable will serve as the primary wire carrying signals between the Mac and the telescope. At the Mac's end, a USB port is the most appropriate way to connect a telescope. This requires an adapter called the Keyspan High Speed USB Serial Adapter to turn the Mac's USB port into a DB9 serial port. The Keyspan adapter is available at http://www.keyspan.com and sells for approximately $30. A third adapter is required to connect the DB9 on the Keyspan with the RJ12 coming from the telescope. That adapter is called, appropriately, an RJ12 to DB9 adapter, and it's available at any local electronics supply store for around $2. Be sure to purchase the kind that is not preassembled because custom wiring will be required to complete the cable.

Wiring the cable correctly isn't difficult, although it does require some careful attention. Very good instructions, including a table of pin-outs and diagrams can be found at http://nineplanets.org/meade/cable.html, and a Google search will turn up other pages with instructions. Cable construction took about an hour, and required a wire cutter, a phone crimper to attach the RJ12 ends to the phone wire, and a bit of manual dexterity and patience for dealing with small conductors.

Under The Night Sky

Once the cable is built, pointing the telescope with Starry Night is simple. The telescope needs to be aligned, either polar or altazimuth, without being connected to the Mac, just as it would be aligned normally. Then plug each end of the cable into its respective port in each device. Start Starry Night, select the **Telescope** tab on the left side of the window, and press the connect button. Starry Night will display a status message indicating the

connection was successful, and a pair of concentric circles representing the telescope's field of view will appear on Starry Night's map. Control-click on an object in the sky, scroll to the bottom of the pop–up menu and select **Slew scope**, and the telescope will begin moving to the selected destination.

The telescope's pointing accuracy will only be as good as the initial alignment, so extra care and time spent perfecting the telescope's orientation is worthwhile. If the telescope happens to get too far off course, Starry Night has a second option in the pop-up menu named **Sync scope**, which will synchronize the computer's field of view representation with the telescope's actual pointing location. Center a known object in the telescope, control-click that object on the map, select **Sync scope**, and everything is back on track.

Telescope control, along with other features such as real–time sky views, an observing log, detailed object information, and the ability to see any object from any point in space at any time, make Starry Night and a Mac powerful tools for the amateur astronomer.

By Aaron Adams

About The Author

Aaron Adams is a LAN Administrator in Dayton, Ohio, and a former star of Apple's "Switch" ad campaign who doesn't get to participate in amateur astronomy as much as he would like because of his day job. He can be reached via e-mail at aaron@aaronadams.net.

Advertiser/Product Index

Introducing the bruAPP™

Plug & Play Mac OS X Network Backup System from TOLIS Group

- D2D + D2D2T Backup Appliance
- 250 GB - 4 TB disk stage NEW
- Scalable to support future network growth
- Remote management via graphical and text consoles

- Fully compatible with Xsan environments
- Support for all SCSI and Fibre-Channel tape devices and libraries
- Built in VXA-2 or LTO-2 tape drives in select models
- Client system support for all major operating systems
- BRU™— 20 years of Backup You Can Trust℠

Mac

One System, One Function, One Solid Performer

5 Minutes From Unpack to First Backup—Literally

Whether you're responsible for a few deskside systems or the latest Hollywood blockbuster, the new bruAPP provides ultra-reliable, easy to use disk-to-disk (D2D) or disk-to-disk-to-tape (D2D2T) network-based backup.

bruAPP complements TOLIS Group's BRU Server for Mac OS X and BRU LE for Mac OS X data backup and restore software products. bruAPP pricing starts at $2,999.

To learn more about the bruAPP and BRU products for Mac OS X, please call 480-505-0488 ext. 252 or visit TOLIS Group on the web at www.tolisgroup.com.

TOLIS Group™

HASP®
SOFTWARE DRM

TLC Banks on HASP Security

"HASP...was the only Software Digital Rights Management solution we would consider... incredibly strong security, so easy to use."

TLC

Pierre Maloka DIRECTOR OF GAME DEVELOPMENT
Mel Liu SOFTWARE TECHNICIAN

TLC Industries is the company behind FlexArcade™, the revolutionary game platform. PowerBall Billiards™, TLC's pool simulation, is quickly becoming an arcade favorite. When software is this hot, it has to be protected by the best. To enable safe, secure delivery to customers, TLC chose HASP, the world's #1 hardware-based software security solution*.

As TLC's Maloka and Liu know, the **HASP Envelope** offers more layers of protection than any software security solution, making it the strongest file wrapping technology available. And the user-friendly interface makes the solution virtually **plug-and-play**. This means TLC can concentrate on developing great games—and leave the security to the security experts.

**IDC Bulletin #31432, 2004.*

Visit Aladdin.com/hasp to view HASP white papers, an online demo, or to request a FREE Developer Evaluation Kit.

For information about TLC or PowerBall Billiards, visit tlcind.com.

Aladdin
SECURING THE GLOBAL VILLAGE

2005 SIIA //CODiE// FINALIST

North America: 1-800-562-2543, 847-818-3800 • **UK** • **Germany** • **Israel** • **Benelux** • **France** • **Spain** • **Asia Pacific** • **Japan**